# DETEKSI PERUNDUNGAN SIBER PADA *TWEET* BERBAHASA INDONESIA MENGGUNAKAN SUPPORT VECTOR MACHINE DENGAN *LEXICON BASED FEATURES*

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Abednego Torang Bolon Panjaitan
NIM: 165150207111083

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

# PENGESAHAN

DETEKSI PERUNDUNGAN SIBER PADA *TWEET* BERBAHASA INDONESIA MENGGUNAKAN SUPPORT VECTOR MACHINE DENGAN *LEXICON BASED FEATURES*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Abednego Torang Bolon Panjaitan
NIM: 165150207111083

Skripsi ini telah diuji dan dinyatakan lulus pada
30 Juni 2021
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
| --- | --- |
| Putra Pandu Adikara, S.Kom., M.Kom. | Dr. Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng. |
| NIP: 19850725 200812 1 002 | NIP: 19840628 201903 1 006 |

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D.
NIP: 19741118 200312 1 002

# PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Balikpapan, 7 Juli 2021


Abednego Torang Bolon Panjaitan

NIM: 165150207111083

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan berkat, rahmat, dan pertolongan-Nya sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul "Deteksi Perundungan Siber pada *Tweet* Berbahasa Indonesia menggunakan Support Vector Machine dengan *Lexicon Based Feature*".

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Putra Pandu Adikara, S.Kom., M.Kom., selaku dosen pembimbing 1 dan Bapak Dr.Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng., selaku dosen Pembimbing 2 yang telah dengan sabar memberikan bimbingan, saran, dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
2. Bapak Adhitya Bhawiyuga, S.Kom., M.Sc. selaku Ketua Program Studi Teknik Informatika,
3. Bapak Achmad Basuki, S.T., M.MG., Ph.D. selaku Ketua Jurusan Teknik Informatika,
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D., selaku dosen Penasihat Akademik yang selalu memberikan nasihat kepada penulis selama menempuh masa studi.
5. Seluruh dosen Fakultas Ilmu Komputer yang telah mendidik dan memberikan ilmu serta wawasannya kepada penulis selama menempuh pendidikan.
6. Bapak Alm. Nasib Panjaitan dan Ibu Friska Simanjuntak selaku orang tua penulis tercinta dan tersayang atas doa, kesabaran, dan segala dukungan yang diberikan demi terselesaikan skripsi ini.
7. Seluruh civitas akademik Fakultas Ilmu Komputer Universitas Brawijaya yang telah menciptakan lingkungan yang baik nan nyaman serta dukungan kepada penulis selama menempuh pendidikan dan menyelesaikan skripsi ini.
8. Rose Diana Rahel Sianturi, S.Psi., selaku psikolog yang membantu penulis menentukan kelas pada dokumen dataset.
9. Teman-teman seperjuangan dan sekontrakan: Andre, David, Faiz, Galang, Ilham, dan Zaky.
10. Teman-teman terdekat penulis yang telah mendorong dan memberikan dukungan secara moral: Josua, Kevin, Erista, Iren, Laura, Demaris, Nisya.
11. Seluruh teman-teman Informatika UB angkatan 2016, serta semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu dan mendukung penulis selama pendidikan sampai terselesaikannya skripsi ini.
12. Tim sepak bola *Manchester United* yang telah mengingatkan penulis agar selalu rendah hati.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan.

Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 16 Juni 2021

Penulis

Abednego Torang Bolon Panjaitan

# ABSTRAK

**Abednego Torang Bolon Panjaitan, Deteksi Perundungan Siber pada *Tweet* Berbahasa Indonesia menggunakan Support Vector Machine dengan *Lexicon Based Features***

**Pembimbing: Putra Pandu Adikara, S.Kom., M.Kom. dan Dr.Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng.**

Salah satu bentuk penyalahgunaan kemudahan interaksi pada sosial media Twitter adalah perundungan siber. Perundungan siber dapat mengakibatkan korban merasa tidak berharga, marah, malu, tidak percaya diri, dan takut. Perundungan yang dapat dilakukan melalui media sosial berupa ejekan, fitnah, dan ancaman. Deteksi Perundungan Siber bertujuan untuk mendeteksi *tweet* yang mengandung perundungan. Support Vector Machine adalah metode yang digunakan dengan menggunakan fitur Term Frequency-Inverse Document Frequency dan tambahan dari fitur Lexicon Based. Data yang digunakan berjumlah 337 data, dengan pembagian komposisi 70% data latih dan 30% data uji. Hasil evaluasi pada sistem menunjukkan bahwa menggunakan *Lexicon Based Feature* memberikan hasil *recall* yang lebih baik daripada hanya menggunakan TF-IDF dan fitur gabungan dari TF-IDF dengan *Lexicon Based Feature*. Hasil evaluasi tertinggi didapatkan dengan menggunakan *Lexicon Based Feature*, yaitu Kernel Linear dengan parameter *lambda* = 0,1, *gamma* = 0,0001, *complexity* = 1, *epsilon* = 0,0001, dan iter max = 10 menghasilkan *accuracy* = 82,69*, recall* = 79,66*, precision* = 88,67, dan *f-measure* = 83,92.

Kata kunci: perundungan siber, Twitter, Support Vector Machine, *Lexicon Based Features*

# ABSTRACT

**Abednego Torang Bolon Panjaitan, *Detection of cyberbullying on Indonesian-language tweets Support Vector Machine with Lexicon Based Features***

***Supervisor:*** **Putra Pandu Adikara, S.Kom., M.Kom.** ***and*** **Dr.Eng. Fitra Abdurrachman Bachtiar, S.T., M.Eng.**

*One form of abuse of the ease of interaction on social media Twitter is cyberbullying. Cyberbullying can cause victims to feel worthless, angry, embarrassed, insecure, and afraid. Bullying that can be done through social media is in the form of flaming, denigration, and threats. Cyber Bullying Detection aims to detect tweets that contain bullying. Support Vector Machine is a method used by using the* Term Frequency-Inverse Document Frequency *feature and the addition of the Lexicon Based Feature. The data used are 337 data, with the composition of 70% training data and 30% test data. The evaluation results on the system show that using Lexicon Based Feature provides better recall results than using only TF-IDF and the combined features of TF-IDF with Lexicon Based Feature. The highest evaluation results were obtained using a Lexicon Based Feature, namely a Linear Kernel with parameters lambda = 0,1, gamma = 0.0001, complexity = 1, epsilon = 0.0001, and iter max = 10 resulting in accuracy = 82.69, recall = 79.66, precision = 88.67, and f-measure = 83.92.*

*Keywords: cyberbullying, Twitter, Support Vector Machine, Lexicon Based Features*

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB 1 PENDAHULUAN

Pendahuluan terdiri dari penjelasan logis mengenai masalah yang dikerjakan dalam penelitian skripsi dan alasan mengapa perlu mencari solusi atas permasalahan tersebut.

## 1.1 Latar Belakang

Kemudahan berinteraksi maupun berekspresi pada jejaring sosial menjadikan banyak orang memilih untuk membagikan perasaan mereka pada platform jejaring sosial yang digunakan. Twitter adalah salah satu platform jejaring sosial yang terkenal karena terdapat beragam pengguna yang berbagi informasi maupun saling sekadar berinteraksi di Twitter. Cara pengguna Twitter berinteraksi satu sama lain adalah dengan membuat *tweet* atau cuitan, yang berarti sebuah unggahan yang dibuat pada jejaring sosial Twitter. Sebuah *tweet* dibatasi dengan maksimal 280 karakter, dan empat media foto atau video. Pada kuartal satu (Januari-Maret) tahun 2019, tercatat pengguna aktif Twitter bulanan menembus angka 330 juta pengguna (Clement, 2019). Pada kuartal empat (Oktober-Desember) tahun 2019, pengguna aktif harian Twitter menembus angka 152 juta (Lunden, 2020). Pengguna aktif Twitter juga tidak sepenuhnya merupakan akun pribadi. Twitter juga digunakan oleh berbagai macam komunitas, perusahaan korporat bahkan beberapa Lembaga pemerintah juga memiliki akun jejaring sosial Twitter.

Penggunaan yang masif serta mudahnya berinteraksi pada Twitter pun tak jarang disalahgunakan oleh beberapa orang. Salah satu bentuk penyalahgunaannya adalah dengan sengaja melontarkan *tweet* negatif yang bersifat untuk menyakiti hati orang yang menerima *tweet* tersebut. Hal tersebut dapat disebut dengan perundungan siber. Perundungan siber adalah salah satu bentuk dari perundungan (*bullying*) yang dilakukan dengan menggunakan media elektronik, terjadi melalui Internet (jejaring sosial) atau ponsel (Sticca & Perren, 2013; Kowalski *et al.*, 2014). Dampak yang ditimbulkan dari perundungan siber pun sangat serius. Pada remaja berusia 12-15 tahun, perundungan siber menekan korban secara mental mengakibatkan korban merasa malu, merasa tidak berharga, marah, tidak bisa konsentrasi belajar, tidak percaya diri dan takut. Contoh perundungan yang dilakukan adalah berupa ejekan, fitnah, dan ancaman. (Sartana & Afriyeni, 2017).

Pada penjelasan Undang-Undang Informasi dan Transaksi Elektronik (UU ITE) nomor 11 tahun 2008 pasal 27 ayat 3 mengenai penghinaan atau pencemaran nama baik telah mengategorikan perundungan siber ke dalamnya karena terdapat penghinaan. KOMINFO pun mengoperasikan mesin sensor untuk memberantas konten negatif, dan perundungan siber termasuk ke dalamnya (Devega, 2017). Berlandaskan permasalahan tersebut, pada pemberantasannya diperlukan adanya sistem otomatis untuk mendeteksi perundungan siber pada *tweet* agar proses penindak lanjutan *tweet* yang termasuk perundungan siber dapat lebih

cepat diatasi, yang diharapkan dapat mengurangi perundungan siber pada sosial media Twitter.

Pada masalah deteksi, terdapat beberapa model yang bisa digunakan untuk mendeteksi *tweet* perundungan siber, yaitu Naive Bayes, Logistic Regression, Stochastic Gradient Descent, K-Nearest Neighbours, Decision Tree, Random Forest, dan Support Vector Machine. Penelitian mengenai deteksi perundungan siber Bahasa Indonesia telah dengan judul "*Cyberbullying Comment Classification on Indonesian Selebgram Using* Support Vector Machine *Method*". Tujuan penelitian tersebut untuk mengklasifikasikan komentar yang mengandung perundungan pada bagian komentar di *Instagram* dengan mengimplementasikan Support Vector Machine dan 1053 komentar yang ditujukan pada selebriti Awkarin, dan didapatkan hasil *accuracy* 79,412%. Pada penelitian ini dijelaskan bahwa *accuracy* dapat ditingkatkan dengan cara mengembangkan fitur semantik pada dataset agar komen yang termasuk satir dapat diklasifikasikan sebagai perundungan (Andriansyah, et al., 2017). Pada kasus perundungan siber telah dilakukan perbandingan metode oleh Kelly Reynolds (2012) dengan membandingkan metode Decision Tree, dengan K-nearest neighbor. Data pada penelitian tersebut diambil pada situs tanya jawab *formspring.me* dan dengan menggunakan data latih yang berjumlah 2696 dan data uji 1219, dan dihasilkan kesimpulan metode Decision Tree menghasilkan nilai *accuracy* tertinggi sebesar 81,7% (Reynolds, et al., 2011). Selanjutnya, enam tahun kemudian penelitian tersebut dilanjutkan kembali oleh Noviantho dengan menggunakan data yang sama yang digunakan oleh Reynolds dan juga membandingkan algoritme Support Vector Machine dengan Naïve Bayes dan menghasilkan nilai *accuracy* tertinggi didapat menggunakan algoritme Support Vector Machine dengan nilai 99,41% (Noviantho, et al., 2017).

Penelitian-penelitian tersebut dapat memberikan kesimpulan bahwa algoritme Support Vector Machine adalah salah satu metode yang baik digunakan dalam pendeteksian perundungan siber, dan untuk meningkatkan kinerja dari metode Support Vector Machine diperlukan kombinasi metode lain. Salah satu metode yang dapat meningkatkan tingkat efisiensi metode Support Vector Machine adalah *Lexicon Based Feature* (Sidiqua, et al., 2016).

Oleh karena itu, berdasarkan analisis yang sudah diuraikan, penulis berharap kombinasi dari model algoritme Support Vector Machine dengan *Lexicon Based Feature* dapat di terapkan ke dalam kasus pendeteksian perundungan siber pada *tweet*, sehingga penulis mengangkat judul "Deteksi Perundungan Siber pada *Tweet* Berbahasa Indonesia Menggunakan Support Vector Machine dengan *Lexicon Based Feature*".

## 1.2 Rumusan Masalah

Didasari penjelasan dari latar belakang, maka dapat ditentukan permasalahan yang dibahas dalam penelitian ini, yaitu:

1. Bagaimana pengaruh penggunaan *Lexicon Based Feature* terhadap hasil evaluasi yang didapatkan?

2. Bagaimana hasil evaluasi metode Support Vector Machine dengan parameter optimal pada pendeteksian perundungan siber?

## 1.3 Tujuan

Berdasarkan perumusan masalah yang telah dijelaskan sebelumnya, maka didapatkan tujuan dari penelitian skripsi ini adalah:

1. Mengetahui pengaruh nilai hasil klasifikasi dengan data yang menerapkan *Lexicon Based Feature* dan data yang tidak menggunakan *Lexicon Based Feature.*
2. Mengetahui hasil *accuracy*, *recall*, *precision*, dan *f-measure* metode Support Vector Machine dengan menggunakan parameter optimal pada pendeteksian perundungan siber.

## 1.4 Manfaat

Manfaat yang bisa di dapat dari penulisan skripsi ini adalah:

1. Membantu pihak pemerintah atau Twitter agar lebih cepat memberantas *tweet* yang bersifat perundungan siber.
2. Menjadi acuan untuk penelitian selanjutnya.

## 1.5 Batasan Masalah

Batasan masalah pada penelitian skripsi ini adalah:

1. Data pada penelitian yaitu berupa *tweet* berbahasa Indonesia yang ditujukan pada Presiden RI bapak Joko Widodo berjumlah 337 *tweet*.
2. Algoritme yang digunakan adalah Support Vector Machine dengan penambahan *Lexicon Based Feature*.

## 1.6 Sistematika Pembahasan

Sistematika penulisan dilakukan guna menjadi dasar penyusun laporan terkait pengerjaan penelitian skripsi yang di dalamnya berisi dari bab-bab, yaitu:

**BAB 1 PENDAHULUAN**

Pendahuluan membahas latar belakang, rumusan masalah, tujuan, manfaat dan batasan masalah mengenai deteksi perundungan siber.

**BAB 2 LANDASAN KEPUSTAKAAN**

Landasan kepustakaan ini merumuskan acuan yang digunakan dalam penelitian. Landasan kepustakaan berisi teori-teori dasar yang berkaitan dengan perundungan siber maupun metode yang digunakan untuk perancangan dan implementasi sistem dan juga berisi kajian pengujian maupun percobaan yang telah dilakukan oleh peneliti lain menggunakan Support Vector Machine untuk klasifikasi sebagai landasan dasar dalam penelitian ini.

**BAB 3 METODOLOGI**

Metodologi ini menguraikan tentang metodologi penelitian yang diimplementasikan, berisi diagram alir metode, Teknik pengumpulan data, dan alur penyelesaian masalah dari implementasi algoritme Support Vector Machine, serta perhitungan *accuracy*.

**BAB 4 PERANCANGAN**

Perancangan ini menguraikan perancangan dari sistem yang dibuat yaitu deteksi perundungan siber dengan menggunakan Support Vector Machine.

**BAB 5 IMPLEMENTASI**

Implementasi ini berisi penjelasan langkah-langkah implementasi sesuai dengan perancangan sistem deteksi perundungan siber dengan menggunakan Support Vector Machine.

**BAB 6 PENGUJIAN DAN ANALISIS**

Pengujian dan analisis ini berisi pengujian kernel dan pengujian parameter dan pengaruh *Lexicon Based Feature* serta analisis nilai evaluasi terbaik dari algoritme Support Vector Machine.

**BAB 7 PENUTUP**

Penutup ini menguraikan kesimpulan yang didasari dari hasil pengujian dan analisis yang telah dilakukan, dan memberikan saran yang berguna bagi penelitian selanjutnya mengenai deteksi perundungan siber pada *tweet* menggunakan Support Vector Machine.

# BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menguraikan mengenai beberapa teori terkait perundungan siber, Twitter, *preprocessing*, perhitungan TF-IDF, *Lexicon Based Feature*, algoritme Support Vector Machine, evaluasi, serta membahas penelitian-penelitian yang telah sebelumnya yang berkaitan dengan implementasi Support Vector Machine.

## 2.1 Kajian Pustaka

Penelitian mengenai perundungan siber telah dilakukan dengan judul "*Cyberbullying Comment Classification on Indonesian Selebgram Using* Support Vector Machine *Method*". Algoritme yang digunakan pada penelitian tersebut adalah Support Vector Machine. Data yang digunakan pada penelitian tersebut diambil pada kolom komentar di akun Instagram Karin Novilda (@awkarin) yang berjumlah 1053 data, dengan jumlah data uji 34 yang diambil secara acak dari data tersebut. Data tersebut dibagi menjadi dua kelas, yaitu perundungan dan bukan perundungan. Dari penelitian tersebut didapatkan hasil *accuracy* sebesar 79,412%. Penulis dari penelitian tersebut memberikan saran bagi peneliti selanjutnya untuk bisa mengimplementasikan fitur semantik agar bisa meningkatkan *accuracy* dari metode Support Vector Machine (Andriansyah, et al., 2017).

Setahun kemudian penelitian mengenai perundungan siber pada platform yang sama juga dilakukan "Analisis Sentimen *Cyberbullying* pada Komentar Instagram dengan Metode Klasifikasi Support Vector Machine". Penelitian tersebut menggunakan algoritme Support Vector Machine. Penelitian tersebut menggunakan 400 data berbentuk komentar pada Instagram dan dibagi menjadi 2 kelas, yaitu sentimen positif perundungan dan sentimen negatif perundungan dan dilakukan melalui empat tahap, yaitu *preprocessing,* perhitungan TF-IDF, implementasi *Lexicon Based Feature*, implementasi algoritme SVM. Kesimpulan dari penelitian tersebut didapatkan hasil evaluasi terbaik dengan *accuracy* sebesar 90%, *precision* dengan nilai 94,44%, *recall* dengan nilai 85%, dan f-*measure* dengan nilai 89,47%. Penelitian ini menggunakan perbandingan komposisi data latih dengan data uji 50:50 dan tidak menggunakan *Lexicon Based Feature* (Luqyana, et al., 2018).

Penelitian pada platform Twitter juga pernah dilakukan penelitian dengan judul "Identifikasi *Tweet Cyberbullying* pada Aplikasi Twitter menggunakan Metode Support Vector Machine dan *Information Gain* (IG) sebagai Seleksi Fitur". Penelitian tersebut menggunakan algoritme Support Vector Machine dan *Information Gain* sebagai seleksi fitur. Kesimpulan yang didapat dari penelitian ini yaitu, hasil seleksi fitur yang diimplementasikan tidak terlalu mengalami peningkatan yang signifikan. Hasil tertinggi yang didapatkan sebelum menggunakan seleksi fitur dengan parameter *Sequential Training* SVM *iter Max* dengan nilai 20, *lambda* (λ) dengan nilai 0,5, *gamma* (γ) 0,001, *epsilon* (ε) dengan nilai 0,000001 dan *complexity* (C) dengan nilai satu adalah *accuracy* dengan nilai 75%, *precision* dengan nilai 70,27%, *recall* dengan nilai 86,66%, dan hasil *f-*

*measure* dengan nilai 77,61%. Sementara itu, hasil tertinggi yang didapatkan setelah menggunakan seleksi fitur *Information Gain* adalah *accuracy* sebesar 76,66%, *precision* sebesar 72,22%, *recall* sebesar 86,66% dan hasil *f-measure* sebesar 78,78%. Penelitian ini memberikan saran bagi peneliti berikutnya untuk menambah jumlah data latih (Purnamasari, et al., 2018).

Dua tahun kemudian juga dilakukan penelitian mengenai perundungan siber pada *tweet* juga dilakukan dengan judul "Deteksi *Cyberbullying* pada Cuitan Sosial Media Twitter". Penelitian tersebut mengumpulkan *tweet* yang di dalamnya terdapat kata-kata perundungan dan tidak terdapat kata-kata perundungan. Penelitian tersebut menggunakan *tweet* berjumlah 1971 yang menjadi data latih kemudian dibagi menjadi 2 kelas dengan rasio 985 *tweet* masuk ke dalam kelas pertama (bukan perundungan) dan kelas 986 *tweet* masuk ke dalam kelas kedua (perundungan). Algoritme yang digunakan pada penelitian tersebut adalah Multinomial Naïve Bayes Classifier, Linear Support Vector Machine, Logistic Regression, dan K-Nearest Neighbor (KNN). Penelitian tersebut menghasilkan kesimpulan bahwa keempat algoritme memiliki performa yang relatif sama, tetapi Linear Support Vector Machine memiliki performa terbaik dengan menghasilkan hasil evaluasi dengan nilai *accuracy*, *precision*, *recall*, dan *F-Measure* tertinggi dengan nilai masing-masing 0.997; 1,00; 1,00; 1,00 (Abdulloh & Hidayatullah, 2019).

Keempat penelitian tersebut mendasari penulis untuk melakukan penelitian mengenai klasifikasi perundungan siber pada *tweet* dengan tujuan dapat memberikan hasil penelitian yang lebih baik dari penelitian yang telah dilakukan sebelumnya.

## 2.2 Perundungan Siber

Perundungan siber adalah salah satu bentuk dari perundungan (*bullying*) yang dilakukan menggunakan media elektronik melalui Internet atau ponsel (Sticca & Perren, 2013; Kowalski *et al.*, 2014). Dampak yang ditimbulkan dari perundungan siber pun sangat serius. Pada remaja berusia 12-15 tahun, perundungan siber mengakibatkan korban merasa malu, merasa tidak berharga marah, tidak bisa konsentrasi belajar, dan takut (Sartana & Afriyeni, 2017). Di Indonesia, persoalan perundungan siber telah dirumuskan pada Undang-Undang Informasi dan Transaksi Elektronik (UU ITE) nomor 11 tahun 2008 pasal 27 ayat 3 yang menjelaskan mengenai penghinaan atau pencemaran nama baik telah dikategorikan sebagai perundungan siber karena terdapat penghinaan. KOMINFO pun mengoperasikan mesin sensor untuk memberantas konten negatif tersebut (Devega, 2017).

Berdasarkan Sartana & Afriyeni (2017), beberapa karakteristik perundungan siber adalah:

1. Mengirimkan pesan bernada kasar atau vulgar, atau biasa disebut dengan *flaming*.

2. Berulang kali mengirim pesan yang bersifat menyerang personal, atau *online harassment.*
3. Mengancam untuk melukai atau mencelakakan atau mengintimidasi secara eksesif.

Contoh *tweet* yang mengandung perundungan siber adalah:

1. @jokowi @basuki_btp @aniesbaswedan Pak Boss.. gimana rasanya sakit hati kala melihat Pak Anies Baswedan penuh Prestasi. Sedangkan Junjunan anda...?? Apa prestasinya..?? NOL BESAR Apalagi si Ahok yg dulu sering ngomong Tai
2. @QaillaAsyiqah Semuga allah segera merobek-robek kekuasaanya @jokowi dan jajarannya.

## 2.3 Twitter

Twitter merupakan salah satu platform jejaring sosial yang terkenal karena terdapat beragam komunitas yang berbagi informasi maupun sekadar saling berinteraksi di Twitter. Cara pengguna Twitter berinteraksi dengan satu sama lain adalah dengan membuat *tweet* atau cuitan, yang berarti sebuah unggahan yang dibuat pada jejaring sosial Twitter. Sebuah *tweet* dibatasi dengan maksimal 280 karakter, dan empat foto atau video (Purnamasari, et al., 2018).

## *2.4 Preprocessing*

*Preprocessing* merupakan sebuah langkah awal untuk mengolah *dataset* agar dapat digunakan menjadi data yang bisa diolah pada proses selanjutnya. Sekumpulan karakter yang bermakna (teks) harus dipecah menjadi unsur yang lebih berarti (Feldman & Sanger, 2007).

### *2.4.1 Cleaning*

*Cleaning* adalah proses menghilangkan *noise* dengan cara menghilangkan karakter yang tidak relevan berupa *hashtag*, simbol, *link*, dan emoji (Gaigole, et al., 2013).

### *2.4.2 Case Folding*

*Case Folding* merupakan proses mengganti semua huruf pada data yang digunakan menjadi huruf kecil (Gaigole, et al., 2013).

### 2.4.3 Tokenisasi

Tokenisasi adalah proses pemisahan *string* menjadi bentuk tunggal. Proses ini digunakan untuk memisahkan setiap kata yang dibelah oleh *whitespace* dan menjadikan mereka sebagai daftar kata/token (Gaigole, et al., 2013).

### *2.4.4 Filtering*

*Filtering* merupakan langkah untuk mengambil kata-kata bermakna dari hasil tokenisasi dengan cara menghapus berbagai kata dalam Bahasa Indonesia yang

sering digunakan tetapi tidak memiliki makna, pada *Information Retrieval* (IR) dan *text mining*, kata tersebut bergelar dengan *Stopword* (Gaigole, et al., 2013).

### *2.4.5 Stemming*

*Stemming* adalah langkah untuk mengubah suatu kata dari hasil tokenisasi menjadi bentuk dasar dari kata tersebut. *Stemming* dilakukan untuk menggabungkan setiap kata yang memiliki akar kata yang sama tetapi dinyatakan dalam padanan yang berbeda, yaitu berupa kata dengan imbuhan (Gaigole, et al., 2013) .

## 2.5 Perhitungan TF-IDF

Data yang telah melewati tahap *preprocessing* harus dapat diolah. Oleh karena itu data tersebut harus dalam bentuk numerik. Pengubahan data menjadi bentuk numerik dapat dilakukan dengan menggunakan bobot Term Frequency–Inverse Document Frequency (*TF*-IDF). Term Frequency merupakan jumlah kemunculan kata/token dalam suatu dokumen atau dapat disebut frekuensi kata dalam satu dokumen (Xia & Chai, 2011). Semakin sering kata tersebut muncul, semakin tinggi Term Frequency dokumen untuk kata tersebut. Sedangkan Term Weighting adalah proses pembobotan pada setiap kata/token. Rumus untuk perhitungan Term Weighting dapat dilihat pada Persamaan 2.1:

$$w_{tf_{t,d}} = \begin{cases} 1 + \log 10\ tf(t,d), & if\ tf(t,d) > 0 \\ 0, & otherwise \end{cases} \quad \textbf{2.1}$$

Keterangan:
$w_{tf_{t,d}}$ : *hasil pembobotan* $tf_{t,d}$
*tf* : Term Frequency
*t* : *term*
*d* : *document*

Document Frequency (DF) merupakan jumlah dokumen yang memiliki suatu token/kata. Sementara Inverse Document Frequency (IDF) adalah hasil inversi dari DF. Rumus untuk perhitungan IDF yang dikemukakan oleh Jones (1972) dapat dilihat pada Persamaan 2.2:

$$idf_t = \log_{10} {}^{N}/_{df_t} \quad \textbf{2.2}$$

Keterangan:
$idf_t$ : *inversi dari* $df_t$
$df_t$ : *jumlah dokumen yang mengandung kata t*
*N* : *jumlah dokumen yang ada*

Bobot TF-IDF dari suatu token/kata didapatkan dari hasil perkalian oleh Term Weighting dengan Inverse Document Frequency (Manning, et al., 2009). Persamaan formulasi TF-IDF telah disajikan dalam Persamaan 2.3:

$$w_{t,d} = w_{tf_{t,d}} * idf_t \quad 2.3$$

Keterangan:

$w_{t,d}$ : *bobot TF-IDF*

$w_{tf_{t,d}}$ : *hasil pembobotan* $tf_{t,d}$

$idf_t$ : *hasil inversi dari* $df_t$

## *2.6 Lexicon Based Feature*

*Lexicon Based Feature* adalah kumpulan kata bersentimen yang digunakan untuk mengindikasi ekspresi negatif atau positif maupun netral pada data. Sentimen tersebut didapatkan dengan cara mencocokkan kata pada data terhadap Kamus *Lexicon* Bahasa Indonesia (Wahid & SN, 2016). Pada Tabel 2.1 dapat dilihat fitur-fitur yang diekstraksi pada penelitian ini.

**Tabel 2.1 Tabel Ekstraksi Fitur**

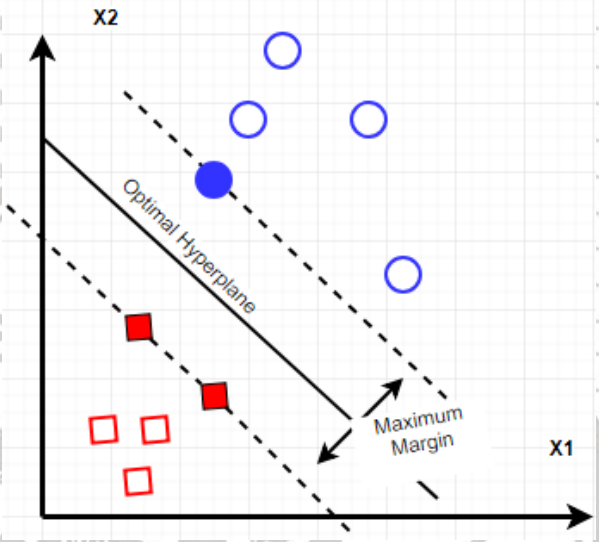
| Fitur | Deskripsi |
|---|---|
| F1 | Jumlah kata penegasan *(booster words)* pada *tweet* |
| F2 | Jumlah kata positif pada *tweet*. |
| F3 | Jumlah kata negatif pada *tweet*. |
| F4 | Jumlah kata positif pada kata sifat *(adjective)*. |
| F5 | Jumlah kata negatif pada kata sifat *(adjective)*. |
| F6 | Jumlah kata positif pada kata kerja *(verb).* |
| F7 | Jumlah kata negatif pada kata kerja *(verb)*. |
| F8 | Jumlah kata positifif pada kata keterangan *(adverb)*. |
| F9 | Jumlah kata negatif pada kata keterangan *(adverb).* |
| F10 | Persentase kata positif pada kata sifat *(adjective)*. |
| F11 | Persentase kata negatif pada kata sifat *(adjective)*. |
| F12 | Persentase kata positif pada kata kerja *(verb).* |
| F13 | Persentase kata negatif pada kata kerja *(verb)*. |
| F14 | Persentase kata positifif pada kata keterangan *(adverb)*. |
| F15 | Persentase kata negatif pada kata keterangan *(adverb).* |

## 2.7 Algoritme Support Vector Machine (SVM)

Support Vector Machine adalah metode yang ditemukan oleh Boser, Guyon, dan Vapnik, muncul pertama kali pada tahun 1992 di *Annual Workshop on*

*Computational Learning Theory*. SVM adalah metode *supervised* yang berkaitan dengan *learning*. Pada masalah klasifikasi teks dengan penggunaan fitur indeks *term* telah dipopulerkan oleh Thorsten Joachim (Luqyana, et al., 2018).

Pada awalnya secara prinsip SVM dirancang untuk menyelesaikan masalah Linear. Klasifikasi pada permasalahan Linear dapat dicari dengan menggunakan *hyperplane* (garis pemisah) (Nugroho, et al., 2003). Ilustrasi dari Support Vector Machine yang menggunakan *hyperplane* untuk memisahkan dua kelompok kelas berbeda telah disajikan pada Gambar 2.1.




**Gambar 2.1 Ilustrasi *Hyperplane***

Pada Gambar 2.1 terlihat jelas bahwa *hyperplane* dapat memaksimalkan jarak pemisah antar kelas pada data, yaitu tiap kelas memiliki pola masing-masing. Karena tidak semua kelas dapat dipisahkan dengan optimal oleh garis Linear, SVM pun kemudian dikembangkan untuk mengatasi masalah non-Linear dengan menggunakan konsep kernel pada vector space dimensi tinggi (Hasanah, et al., 2019). Kernel pada SVM memiliki berbagai jenis beberapa di antaranya yaitu, Linear, Polynomial of Degree d, Gaussian RBF, dll.

Penelitian ini menggunakan Kernel Gaussian RBF yang memiliki rumus fungsi persamaan yang telah disajikan pada Persamaan 2.4.

$$\boldsymbol{K(X,Y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)} \quad \textbf{2.4}$$

Keterangan:

$K(X,Y)$ : *fungsi kernel*

$X - Y$ : *fitur*

$\sigma$ : *sigma*

Pada data latih digunakan algoritme *Sequential Training* yang merupakan algoritme yang bertujuan untuk menempatkan titik data dengan jarak terdekat pada *hyperplane* (Vijayakumar & Wu, 1999). Tahap perhitungan *sequential learning* dilakukan sebagai berikut:

1. Menginisialisasi parameter *alpha* untuk mencari Support Vector (α), epsilon untuk perhitungan *error* (ε), *gamma* untuk mengontrol kecepatan (γ) dan konstanta (C).
2. Menghitung matriks Hessian dengan menggunakan rumus pada Persamaan 2.5.

$$\boldsymbol{D_{i,j} = y_i y_j (k(x_i, x) + \lambda^2)} \quad \textbf{2.5}$$

Keterangan:

$D_{i,j}$ : *matriks Hessian*

$y_i$ : *kelas dari data ke i*

$y_j$ : *kelas dari data ke j*

$k(x_i, x)$ : *fungsi kernel*

3. Melakukan perulangan dari data ke-*i* hingga data ke-*j* dengan menggunakan rumus persamaan berikut

a. Cari *error rate* dengan rumus pada Persamaan 2.6.

$$\boldsymbol{E_i = \sum_j^i \alpha_j D_{ij}} \quad \textbf{2.6}$$

Keterangan:

$E_i$ : *error rate*

$\alpha_j$ : *alpha ke j*

$D_{ij}$ : *matriks Hessian*

b. Hitung nilai delta alpha dengan Persamaan 2.7.

$$\boldsymbol{\delta\alpha_i = min\{max[\gamma(1 - E_i), -\alpha_i], C - \alpha_i\}} \quad \textbf{2.7}$$

Keterangan:

$\delta\alpha_i$ : *delta alpha*

$\alpha_i$ : *alpha ke i*

$\gamma$ : *gamma*

$E_i$ : *error rate*

$C$ : *Complexity*

c. Perbarui nilai alpha baru dengan Persamaan 2.8.

$$\boldsymbol{\alpha_i = \alpha_i + \delta\alpha_i} \quad \textbf{2.8}$$

Keterangan:

$\alpha_i$ : *alpha ke i*

$\delta\alpha_i$ : *delta alpha*

4. Lakukan langkah ketiga sampai mencapai batas maksimum iterasi atau ketika $(|\delta\alpha_i|) < \varepsilon(epsilon)$.

5. Perhitungan di atas akan menghasilkan nilai Support Vector. Adapun persamaan Support Vector terdapat pada Persamaan 2.9.

$$\boldsymbol{SV = \alpha_i} \quad \textbf{2.9}$$

Keterangan:

$\alpha_j$ : *alpha ke i*

$SV$ : *support vector*

### *2.7.2* Pengujian Support Vector Machine

Pengujian dilakukan dengan mencari nilai $f(x)$ untuk mencari kelas dari data uji. Penghitungan nilai $f(x)$ pertama membutuhkan nilai bias. Rumus perhitungan pada nilai bias menggunakan Persamaan 2.10.

$$\boldsymbol{b = -\frac{1}{2}\left[\sum_{i=1}^{n}(w * x^{+}) + \sum_{i=0}^{n}(w * x^{-})\right]} \quad \textbf{2.10}$$

Keterangan:

$b$ : *bias*

$w$ : *nilai bobot*

$K(x_i, x)$ : *fungsi kernel*

$n$ : *jumlah data*

Setelah mendapatkan nilai dari bias (b), dilakukan proses perhitungan $f(x)$ dengan menggunakan Persamaan 2.11.

$$\boldsymbol{f(x) = \sum_{i=0}^{n}\alpha_i y_i K(x_i, x) + b} \quad \textbf{2.11}$$

Keterangan:

$\alpha_j$ : *alpha ke i*

$y_i$ : *kelas data latih ke i*

$K(x_i, x)$ : *fungsi kernel*

$b$ : *bias*

$n$ : *banyaknya data*

### 2.7.3 Kelebihan Support Vector Machine

Beberapa kelebihan dari algoritme Support Vector Machine adalah:

1. Memberikan *error* generalisasi yang relatif lebih kecil.
2. Menghasilkan model klasifikasi yang baik walaupun menggunakan data yang sedikit.
3. Mudah diimplementasikan karena masalah dapat dirumuskan.

### 2.7.4 Kelemahan Support Vector Machine

1. Sulit diimplementasikan pada masalah berskala besar
2. SVM umumnya menyelesaikan masalah klasifikasi dengan dua kelas

## 2.8 Evaluasi

Oleh karena model yang digunakan oleh Support Vector Machine adalah *supervised learning*, untuk melakukan evaluasi dari algoritme Support Vector Machine dilakukan dengan menghitung nilai dari *recall*, *precision, F-Measure*, dan *accuracy*. Untuk menghitung *recall*, *precision, F-Measure*, dan *accuracy*, diperlukan adanya *confusion matrix*. *Confusion matrix* adalah salah satu cara untuk mendapatkan nilai dari *true positive, true negative, false negative,* dan *false positive* dalam sebuah penelitian. Nilai evaluasi dalam tabel *confusion matrix* didapat dengan melakukan perbandingan kelas dari hasil aktual dengan kelas yang didapat dari hasil prediksi (Manning, et al., 2009). Nilai dan tabel *confusion matrix* disajikan pada Tabel 2.2.

**Tabel 2.2 *Confusion Matrix***

| Hasil Aktual | Hasil Prediksi | |
|---|---|---|
| | Perundungan | Bukan perundungan |
| Perundungan | TP | FN |
| Bukan perundungan | FP | TN |

Keterangan:

$TP$ : *true positive*

$FP$ : *false positive*

$FN$ : *false negatif*

$TN$ : *true negatif*

### 2.8.1 Precision

*Precision* adalah ketepatan dari hasil yang diperoleh dan relevan (Putri, et al., 2013). Rumus persamaan *precision* dapat dilihat pada Persamaan 2.12.

$$\boldsymbol{Precision = \frac{TP}{TP+FP}} \quad \textbf{2.12}$$

### 2.8.2 Recall

*Recall* adalah kemampuan untuk memperoleh semua dokumen relevan (Putri, et al., 2013). Rumus persamaan *recall* dapat dilihat pada Persamaan 2.13.

$$\boldsymbol{Recall = \frac{TP}{TP+FN}} \quad \textbf{2.13}$$

### 2.8.3 F-Measure

*F-Measure* merupakan nilai kinerja gabungan *precision* dan *recall* (Herdiawan, 2016). Rumus persamaan *precision* dapat dilihat pada Persamaan 2.14.

$$\boldsymbol{F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}} \quad \textbf{2.14}$$

### 2.8.4 Accuracy

*Accuracy* merupakan derajat kedekatan antara hasil prediksi dengan hasil aktual (Herdiawan, 2016). Rumus persamaan *accuracy* telah disajikan pada Persamaan 2.15.

$$\boldsymbol{Accuracy = \frac{TP+TN}{TP+FN+FP+TN} * 100\%} \quad \textbf{2.15}$$

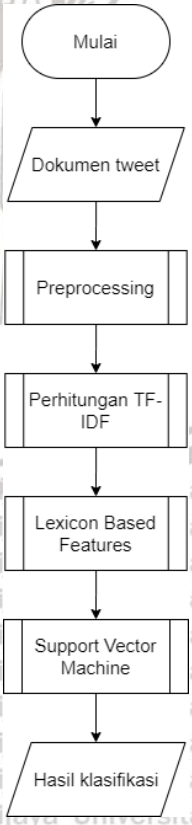
# BAB 3 METODOLOGI

## 3.1 Tipe Penelitian

Tipe penelitian pada skripsi ini merupakan non implementatif analitik. Penelitian bertipe non implementatif analitik merupakan penelitian yang bertujuan mencari hubungan antar tiap elemen pada objek penelitian yang dipakai sebagai akar dalam mengambil keputusan.

## 3.2 Metode Penelitian

Terdapat lima tahap yang dilakukan terhadap penelitian ini. Tahap pertama yaitu memasukkan *input* berupa dokumen. Dokumen dikumpulkan dengan mengambil dokumen berupa *tweet* yang ditujukan pada Presiden RI bapak Joko Widodo, yang berjumlah 337 *tweet*. Setelah itu dilanjutkan dengan tahap *preprocessing* yang melibatkan lima tahap, yaitu *cleaning, case folding,* tokenisasi, *filtering,* dan *stemming*, kemudian dilakukan perhitungan TF-IDF. Setelah perhitungan didapatkan, dilanjutkan dengan pengimplementasian Support Vector Machine. Terakhir sistem menghasilkan *output* berupa hasil klasifikasi dari algoritme Support Vector Machine dengan hasil perhitungan *recall*, *precision*, *F-Measure,* dan *accuracy*. Diagram alir metode telah disajikan dalam Gambar 3.1 Diagram Alir Metode.




**Gambar 3.1 Diagram Alir Metode**

Tahap-tahap penyelesaian masalah dari metode yang diusulkan penulis adalah sebagai berikut:

**1.** *Input* Dokumen *tweet*

Memasukkan data yang berupa *tweet* yang dipakai sebagai data latih dan data uji.

***2.*** *Preprocessing*

Proses pengolahan atau *preprocessing* dilakukan terhadap dokumen untuk mengubah data menjadi bentuk mudah diolah agar data dapat dianalisis untuk proses perhitungan selanjutnya. Tahap-tahap *preprocessing* pada penelitian ini adalah *cleaning*, *case folding*, tokenisasi, *filtering/stopword removal,* dan *stemming*.

**3.** Perhitungan TF-IDF

Pembobotan TF-IDF dilakukan kepada data yang telah melewati tahap *preprocessing*. Inverse Document Frequency dapat mendeteksi keunikan kata dalam kumpulan dokumen, persamaan yang telah di jabarkan dalam Persamaan 2.1 hingga Persamaan 2.3 menjadi acuan dalam perhitungan TF-IDF.

4. Penerapan *Lexicon Based Feature*

Setelah melakukan pembobotan TF-IDF, dilakukan penambahan bobot fitur dengan menerapkan *Lexicon Based Feature* melalui pencocokan dokumen *tweet* pada Kamus *Lexicon* Bahasa Indonesia yang mereferensikan Tabel 2.1 sebagai fitur yang digunakan.

**5.** Proses Klasifikasi Support Vector Machine

Setelah mendapatkan bobot fitur TF IDF dan *Lexicon*, proses dilanjutkan dengan perhitungan Kernel *Gaussian RBF* dengan menggunakan Persamaan 2.4, dan ordo pada penelitian ini adalah dua. Setelah mendapatkan nilai dari fungsi kernel, tahapan berikutnya adalah melakukan implementasi dari *Sequential Training* pada data latih dengan tahapan, menginisialisasi parameter *alpha* (α), *epsilon* (ε), *gamma* (γ) dan konstanta (C). Kemudian menghitung matriks Hessian dengan menggunakan Persamaan 2.5. Kemudian perhitungan dilanjutkan dengan melakukan perulangan nilai *error rate* dengan rumus Persamaan 2.6, nilai delta alpha dengan rumus Persamaan 2.7, dan nilai alpha dengan rumus Persamaan 2.8. Selepas melakukan *Sequential Training* pada data uji, dilakukan perhitungan nilai bias dengan menggunakan Persamaan 2.10. Kemudian setelah mendapatkan hasil dari semua variabel yang dibutuhkan, langkah selanjutnya adalah mengetahui hasil kelas dari data dengan menggunakan Persamaan 2.10.

**6.** Hasil Klasifikasi

Setelah melalui semua proses tahapan yang telah dijabarkan, didapatkan kategori terhadap *tweet* yang didasari hasil dari rumus Persamaan 2.10. Jika hasil tersebut positif, kelas data tersebut adalah bukan perundungan siber. Namun, jika hasil perhitungan negatif, kelas tersebut termasuk perundungan siber.

## 3.3 Teknik Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan dokumen berupa *tweet*. Data berisi dokumen yang berjumlah 337 *tweet* berbahasa Indonesia yang ditujukan kepada Presiden RI bapak Joko Widodo dengan akun Twitter @jokowi. Data yang dipakai diperoleh dengan menggunakan API Twitter dan memakai Bahasa pemrograman Python.

# BAB 4 PERANCANGAN

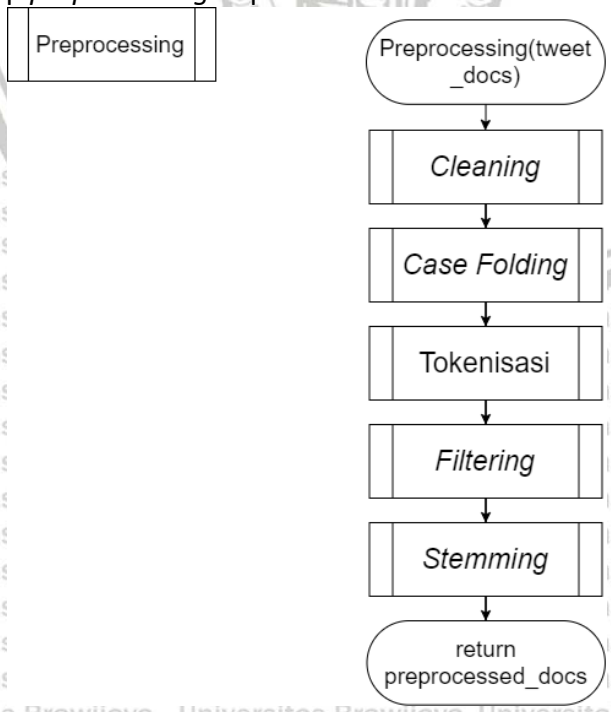
Bab perancangan menjelaskan tentang langkah untuk mencari solusi pada masalah dalam penelitian. Langkah-langkah tersebut yaitu, formulasi masalah, penjelasan metodologi, dan manualisasi.

## 4.1 Formulasi Permasalahan

Deteksi perundungan siber ini dibuat untuk keperluan mendeteksi perundungan siber (anomali) dalam *tweet*. Pada permasalahan ini, terdapat beberapa fitur yang berpengaruh dalam mendeteksi *tweet* yang mengandung anomali. Fitur yang memengaruhi tersebut adalah pembobotan TF-IDF ditambah dengan *Lexicon Based Feature* yang terdiri dari 15 fitur yang telah di jabarkan pada Tabel 2.1. Kemudian, dari beberapa fitur yang memengaruhi tersebut dilakukan proses pendeteksian menggunakan Support Vector Machine. *Output* yang dihasilkan berupa kelas terhadap data uji yang dibagi menjadi dua kelas, yaitu kelas yang mengandung perundungan (negatif) dan kelas yang tidak mengandung perundungan (positif). Langkah-langkah pada penelitian ini diuraikan pada diagram di bawah.

## 4.2 *Preprocessing*

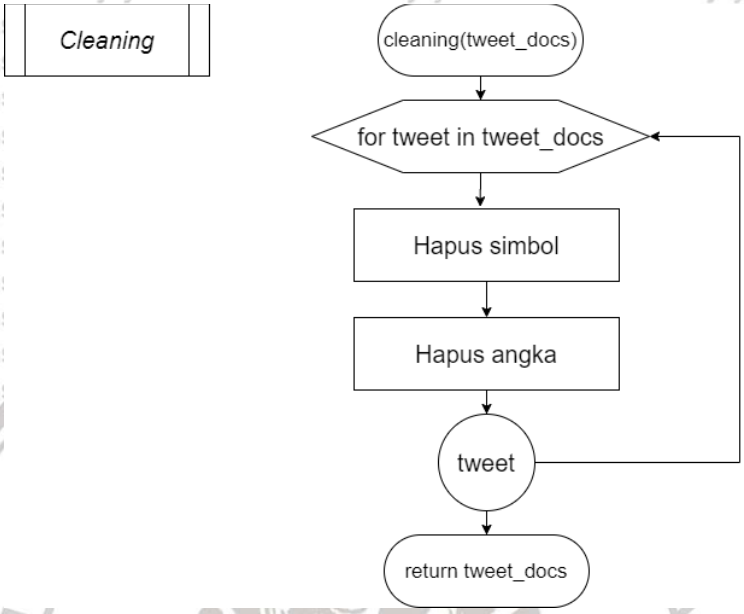
*Preprocessing* merupakan langkah awal pada penelitian ini. Tahap ini mencakup *cleaning*, *case folding,* tokenisasi, *filtering*, dan *stemming*. Diagram alir tahap *preprocessing* dapat dilihat dalam Gambar 4.1.




**Gambar 4.1 Diagram Alir *Preprocessing***

### 4.2.1 Cleaning

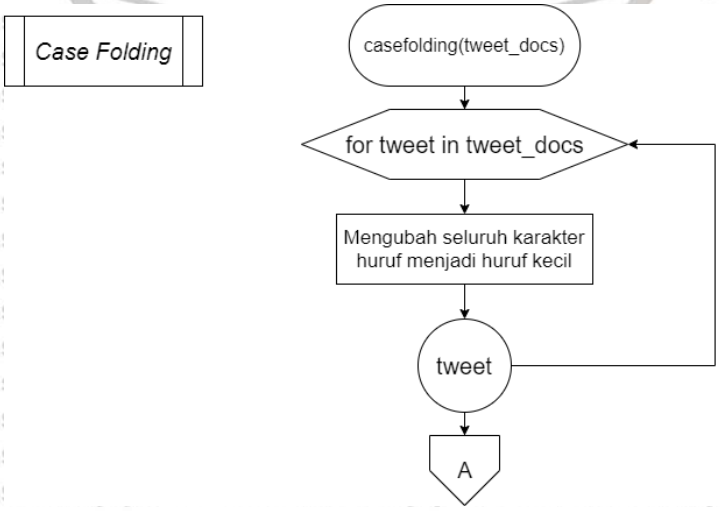
*Cleaning* adalah tahap awal pada *preprocessing.* Proses ini bertujuan untuk menghapus semua simbol maupun angka pada dokumen. Pada tahapan ini digunakan data berupa *tweet* sebagai masukan. *Output* dari tahap ini adalah *tweet* tanpa simbol maupun angka, yang kemudian digunakan pada tahap *preprocessing*. Diagram alir proses *cleaning* telah diuraikan pada Gambar 4.2.

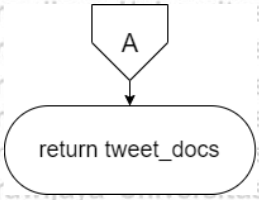



**Gambar 4.2 Diagram Alir *Cleaning***

### 4.2.2 Case Folding

Tahap setelah proses *cleaning* adalah tahap *case folding*. Tahapan ini bertujuan untuk mengganti isi dari semua dokumen menjadi huruf kecil. Proses ini menggunakan dokumen *tweet* yang telah melewati proses *cleaning* menjadi masukan. *Output* dari tahap ini adalah dokumen *tweet* yang tidak memiliki huruf kapital. Alur dari proses *case folding* dapat dilihat pada Gambar 4.3.

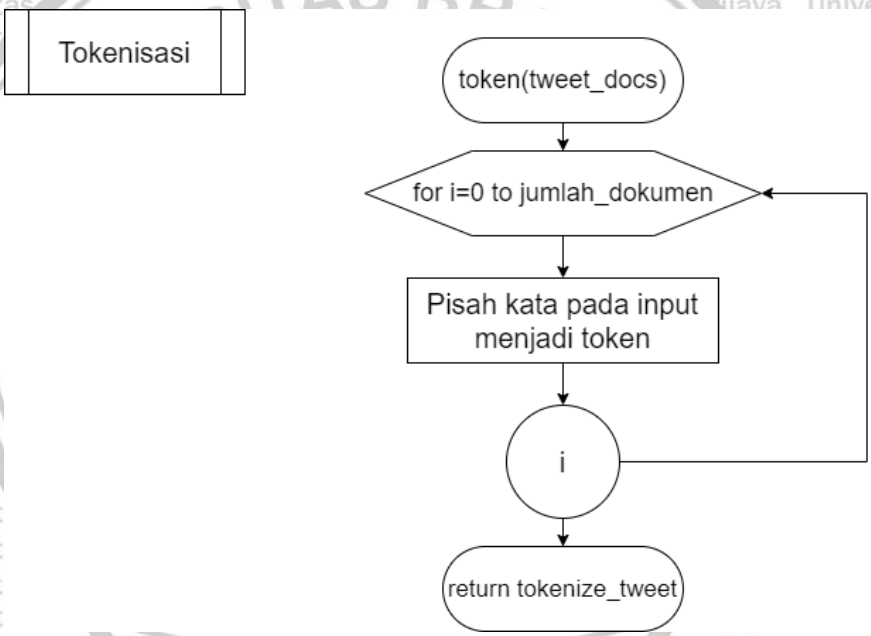





**Gambar 4.3 Diagram Alir *Case Folding***

### 4.2.3 Tokenisasi

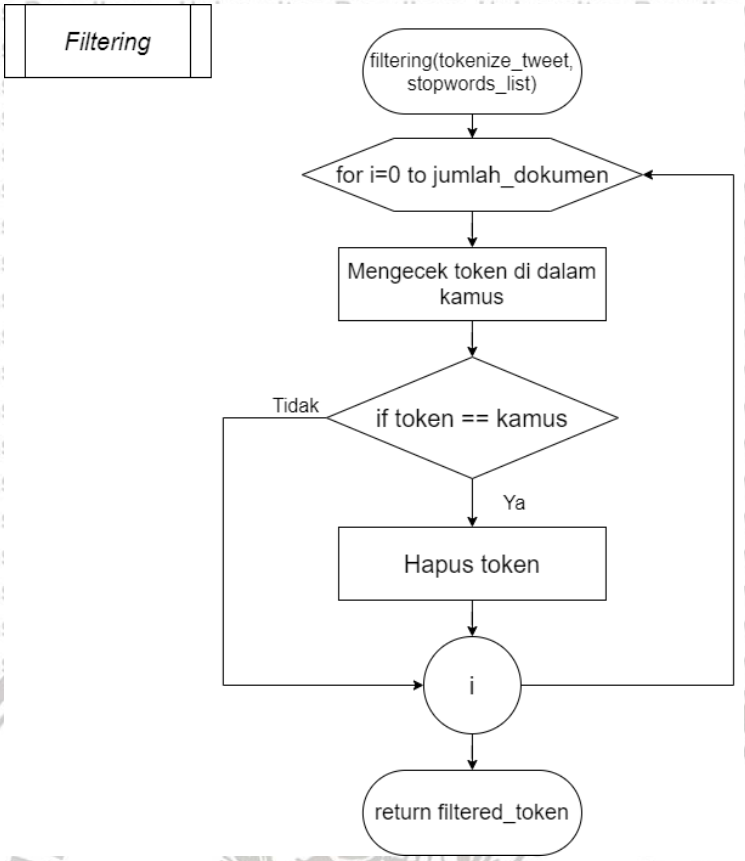
Tokenisasi adalah tahap ketiga pada tahap *preprocessing*. Tahapan ini dilakukan untuk mengekstraksi kata/token pada dokumen *tweet* dengan cara mengambil semua teks yang dipisahkan oleh *blank space* atau spasi. Proses ini menggunakan dokumen *tweet* yang telah melewati proses *case folding* sebagai masukan. *Output* pada tahap tokenisasi adalah *string*/kata dari dokumen *tweet* yang telah dipisahkan. Alur proses tokenisasi dapat dilihat pada Gambar 4.4.




**Gambar 4.4 Diagram Alir Tokenisasi**

### *4.2.4 Filtering*

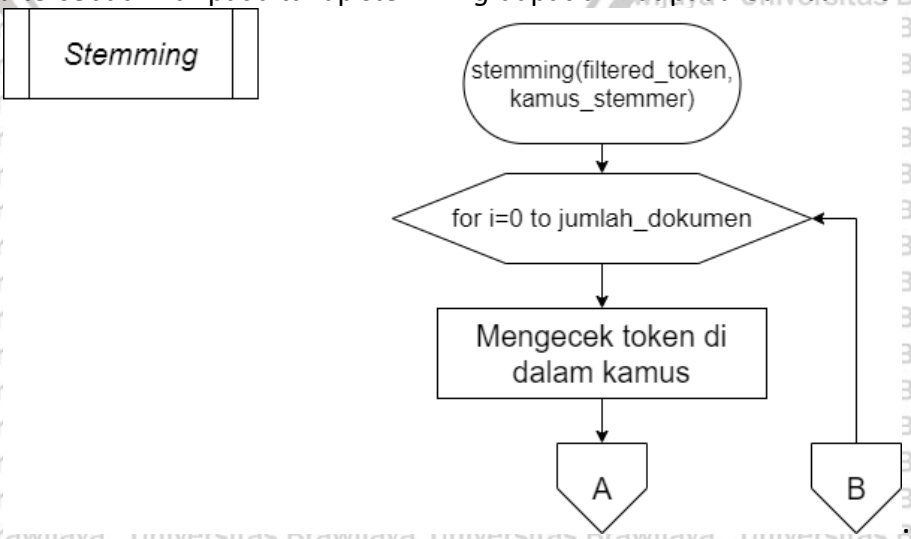
Setelah mengubah data menjadi token, tahapan selanjutnya adalah *filtering*. *Filtering* bertujuan untuk menghapus *stopwords* pada token dan menyisakan token-token yang mengandung makna penting. Proses ini menggunakan dokumen *tweet* yang sudah menjadi token, dan kamus *stopwords* berbahasa Indonesia sebagai masukan. *Output* dari proses ini berupa token-token tanpa *stopword*. Alur pada tahap *filtering* dapat dilihat pada Gambar 4.5.

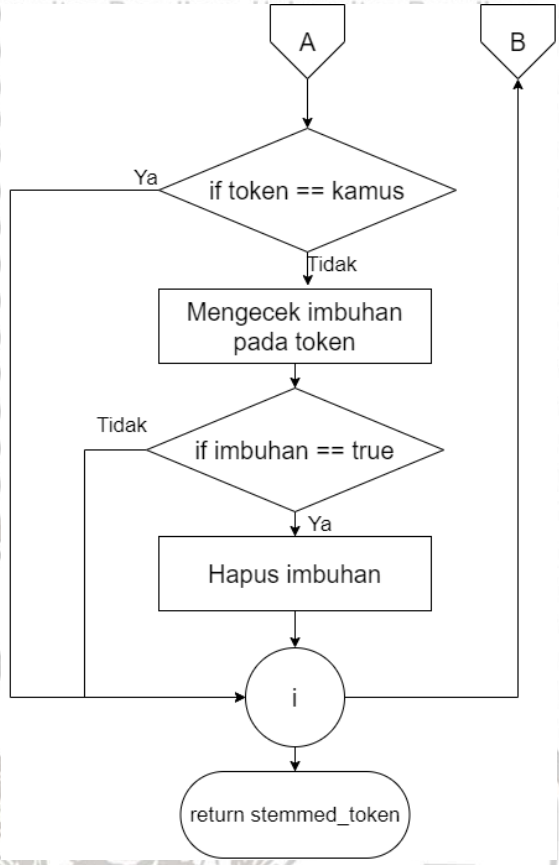



**Gambar 4.5 Diagram Alir Filtering**

### *4.2.5 Stemming*

Tahap dalam proses *preprocessing* yang terakhir adalah *stemming*, yaitu langkah untuk mengubah semua kata menjadi kata dasarnya dengan cara menghilangkan imbuhan dalam kata tersebut. Proses ini memakai dokumen *tweet* yang telah melewati proses *filtering* sebagai masukan dan kamus imbuhan yang digunakan sebagai aturan dalam menghapus imbuhan yang ada pada token. *Output* dari proses ini berupa token-token tanpa imbuhan atau hanya sebatas akar dari kata tersebut. Alur pada tahap *stemming* dapat dilihat pada Gambar 4.6

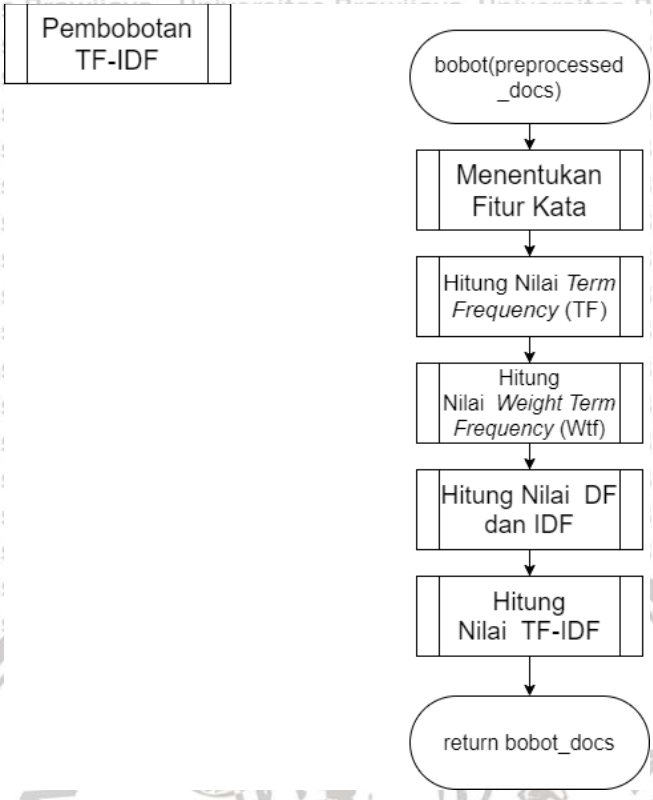





**Gambar 4.6 Diagram Alir Stemming**

## 4.3 Pembobotan kata (TF-IDF)

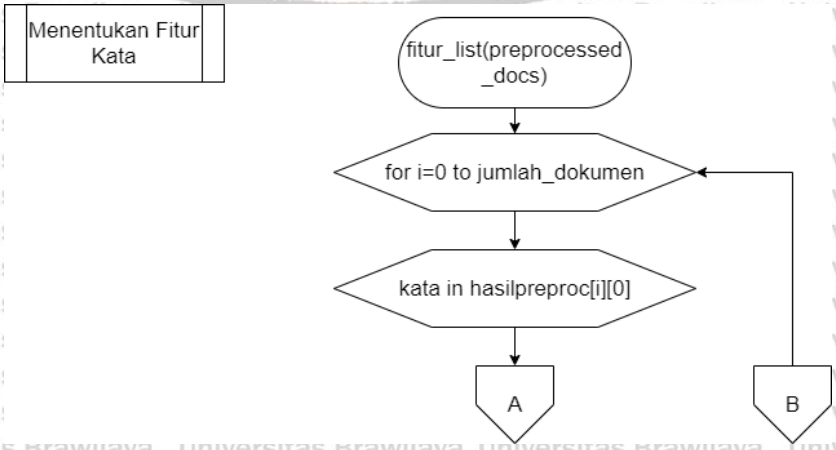
Setelah dilakukan *preprocessing* terhadap data, tahapan berikutnya merupakan pemberian bobot kata/token dengan memakai TF-IDF guna mengubah token menjadi bentuk numerik atau fitur untuk dapat diolah oleh algoritme SVM. Tahapan dalam pembobotan TF-IDF dimulai dari menentukan fitur kata yang berasal token dari dokumen yang telah melewati proses *preprocessing.* Kemudian pada fitur tersebut dilakukan perhitungan nilai dari Term Frequency (TF) atau jumlah kemunculan tiap kata pada *tweet*. Lalu menghitung *Weight* Term Frequency ($W_{tf}$) atau bobot dari Term Frequency yang dihitung dengan menggunakan formula yang telah dirumuskan dalam Persamaan 2.1. Setelah itu menentukan Inverse Document Frequency (IDF) dengan cara menentukan Document Frequency yaitu jumlah kemunculan kata pada suatu dokumen terlebih dahulu kemudian melakukan perhitungan dengan menggunakan rumus pada yang telah dijabarkan dalam Persamaan 2.2. Kemudian mendapatkan nilai dari $W_{tf}$ dan IDF maka kita dapat melakukan langkah terakhir yaitu perhitungan bobot kata TF-IDF dengan menggunakan rumus dalam Persamaan 2.3. Diagram alir pada pembobotan TF-IDF dapat dilihat dalam Gambar 4.7.

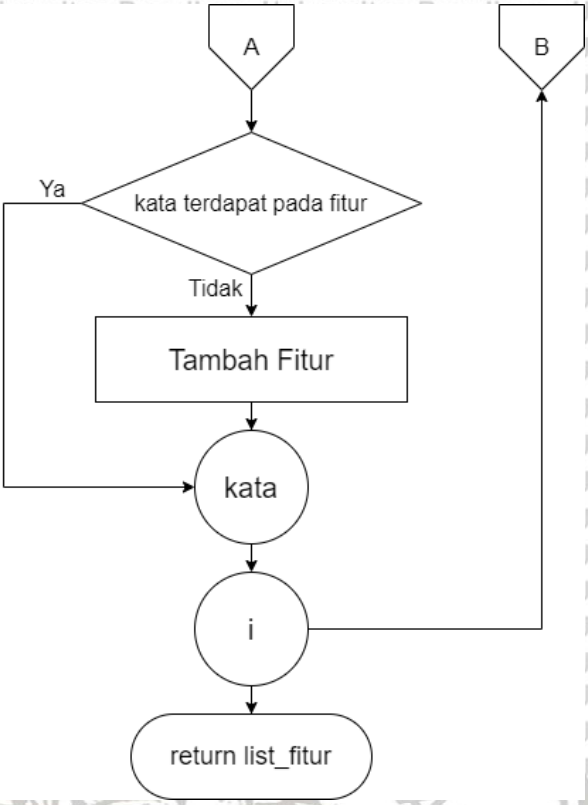



**Gambar 4.7 Diagram Alir Pembobotan Kata (TF-IDF)**

### 4.3.1 Menentukan Fitur Kata

Langkah pertama dalam menentukan bobot kata TF-IDF adalah menentukan fitur kata yang digunakan. Fitur kata yang digunakan didapat dari hasil dari tahap *preprocessing* yang telah dilakukan sebelumnya. Untuk melakukan tahap penentuan fitur kata diperlukan masukan berupa dokumen yang telah melewati tahap *preprocessing* kemudian dilakukan seleksi kondisi *if-else* dengan kondisi apabila kata sudah terdapat pada token, maka dilanjut pada kata selanjutnya. Proses ini menghasilkan *Output* daftar fitur kata dari semua dokumen. Diagram alir dalam menentukan fitur kata dapat dilihat dalam Gambar 4.8.

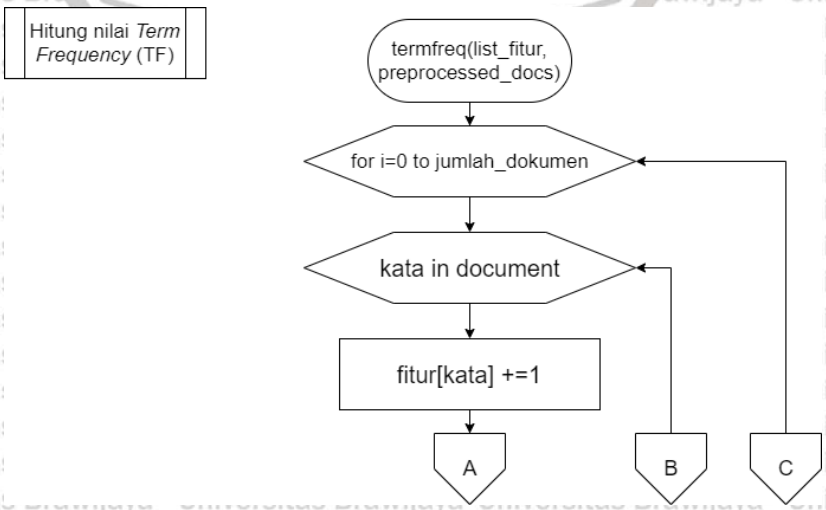


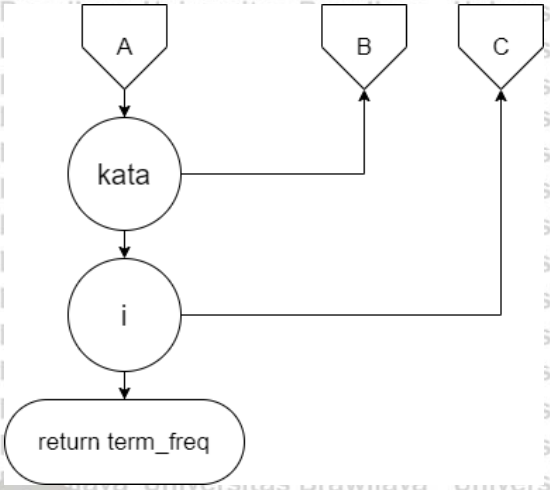



**Gambar 4.8 Diagram Alir Menentukan Fitur Kata**

### 4.3.2 Hitung Nilai Term Frequency (TF)

Tahapan selanjutnya adalah melakukan perhitungan Term Frequency (TF) terhadap fitur dengan cara menghitung jumlah kemunculan suatu token pada dokumen. Masukan pada proses ini adalah daftar fitur yang didapatkan setelah melakukan proses penentuan fitur kata, dan dokumen *tweet* yang telah melewati tahap *preprocessing. Output* dari proses ini adalah nilai TF pada tiap fitur dari setiap dokumen. Diagram alir dari perhitungan nilai TF dapat dilihat dalam Gambar 4.9.

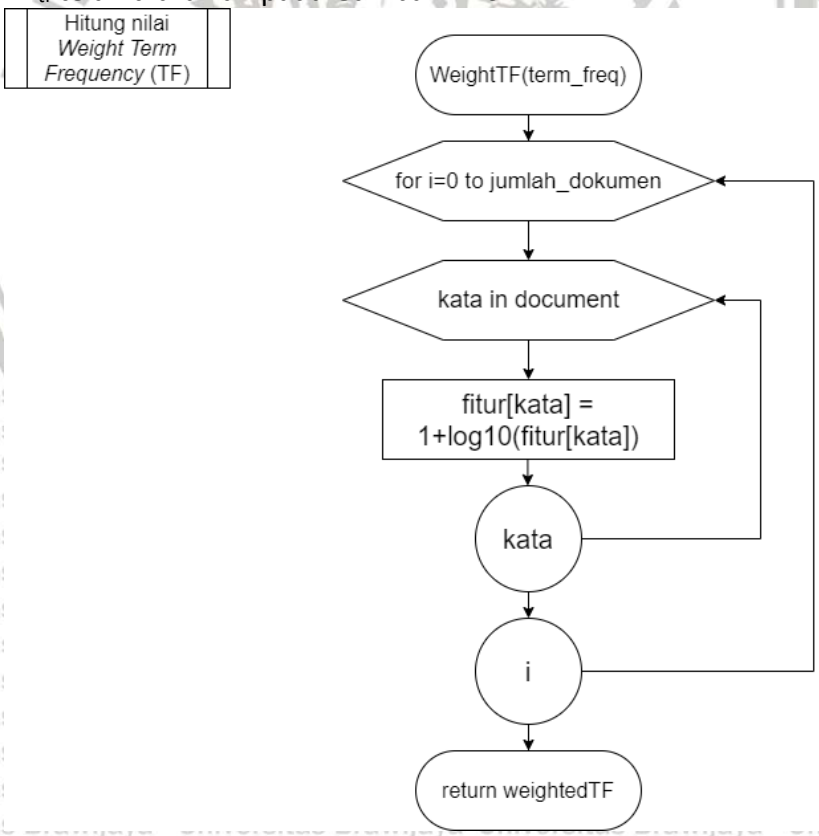





**Gambar 4.9 Diagram Alir Perhitungan Term Frequency (TF)**

### 4.3.3 Hitung Nilai $W_{tf}$

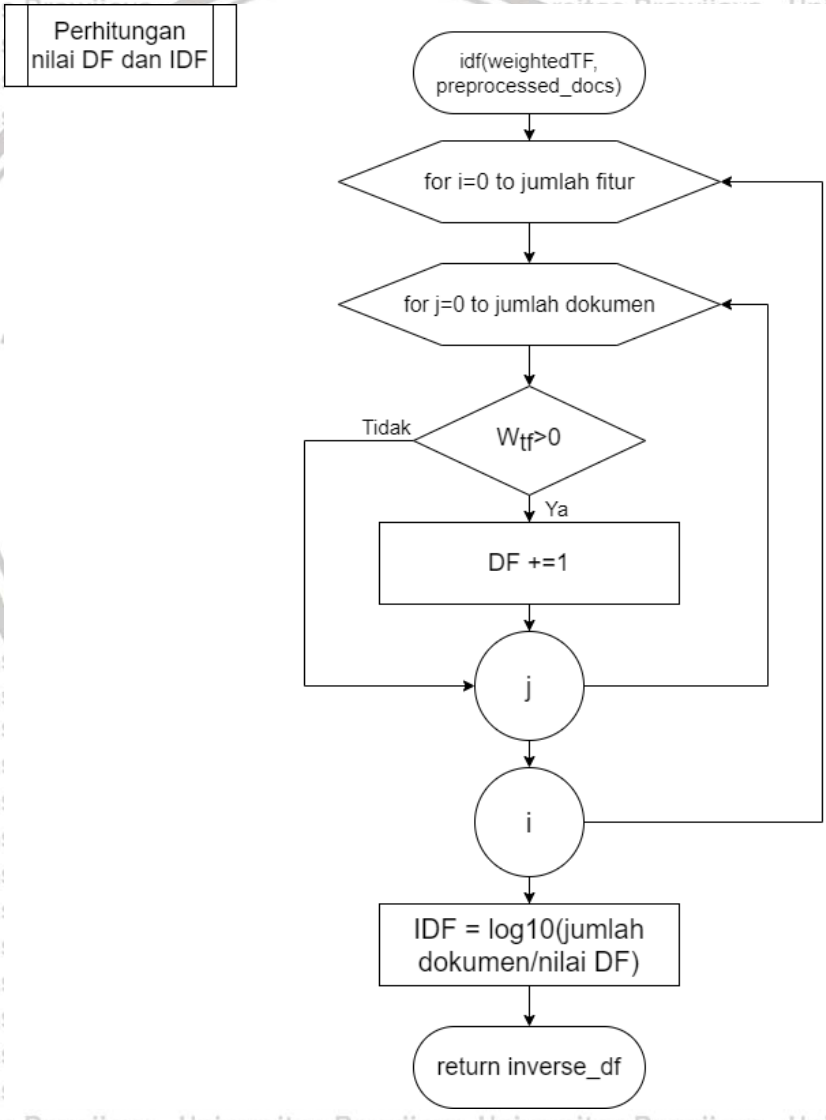
Tahap selanjutnya adalah melakukan perhitungan nilai $W_{tf}$ atau pembobotan dari Term Frequency dengan melakukan perhitungan menggunakan rumus dari Persamaan 2.1. Masukan dari proses ini adalah nilai Term Frequency dari tiap kata. *Output* dari tahap ini adalah nilai $W_{tf}$ dari tiap kata. Diagram Alir dari perhitungan nilai $W_{tf}$ telah diuraikan pada Gambar 4.10.




**Gambar 4.10 Diagram Alir Perhitungan Nilai Wtf**

### 4.3.4 Hitung Nilai DF dan IDF

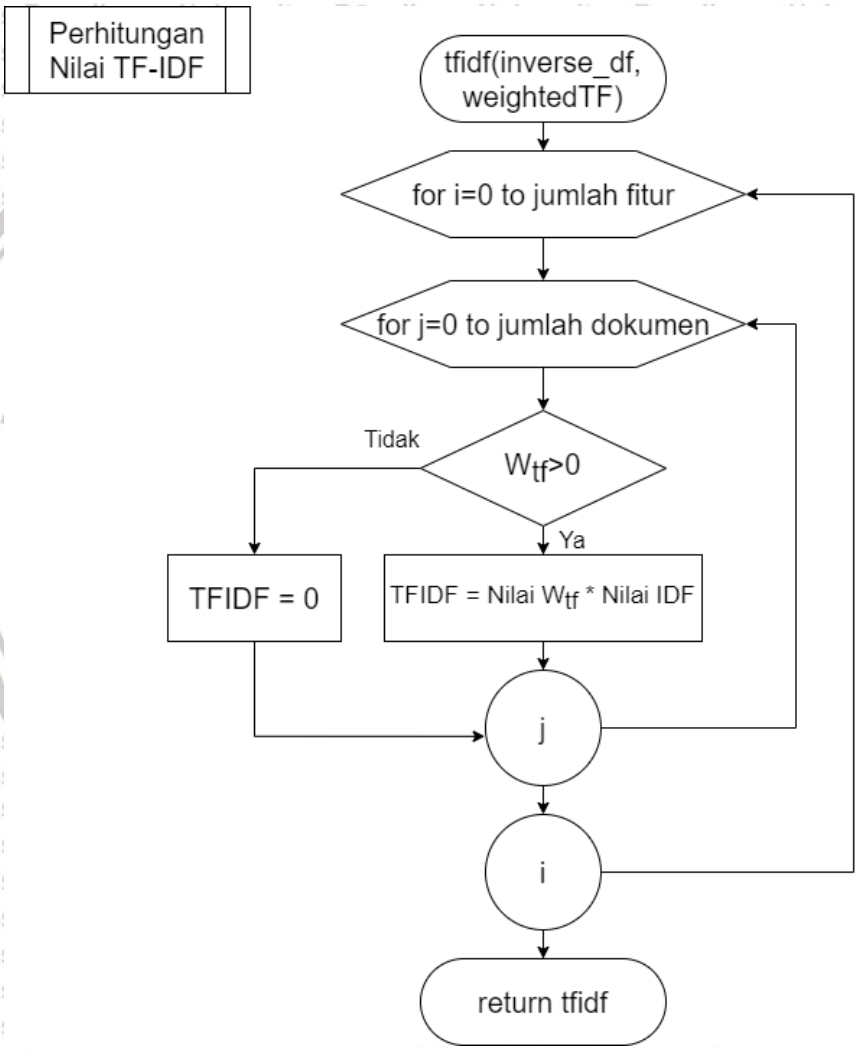
Tahap selanjutnya adalah menentukan Document Frequency (DF) dan Inverse Document Frequency (IDF). Tahapan ini dilewati dengan melakukan perhitungan menggunakan rumus pada Persamaan 2.2. Untuk menghitung nilai DF diperlukan data kemunculan kata pada dokumen, hal tersebut bisa didapatkan melalui perhitungan $W_{tf}$. Setelah mendapatkan nilai DF maka kita dapat menghitung nilai IDF, semakin besar nilai dari DF maka akan semakin kecil nilai dari IDF, sedangkan semakin kecil nilai DF maka akan semakin besar nilai IDF. Oleh karena itu apabila ada kata yang muncul pada semua dokumen maka nilai dari IDF-nya akan menjadi nol. Masukan pada proses ini adalah nilai dari $W_{tf}$ dan juga data dokumen yang telah melewati tahapan *preprocessing*. *Output* pada proses ini adalah nilai dari IDF pada tiap fitur kata. Diagram alir perhitungan DF dan IDF telah disajikan pada Gambar 4.11.




**Gambar 4.11 Diagram Alir Perhitungan Nilai DF dan IDF**

### 4.3.5 Hitung Nilai TF-IDF

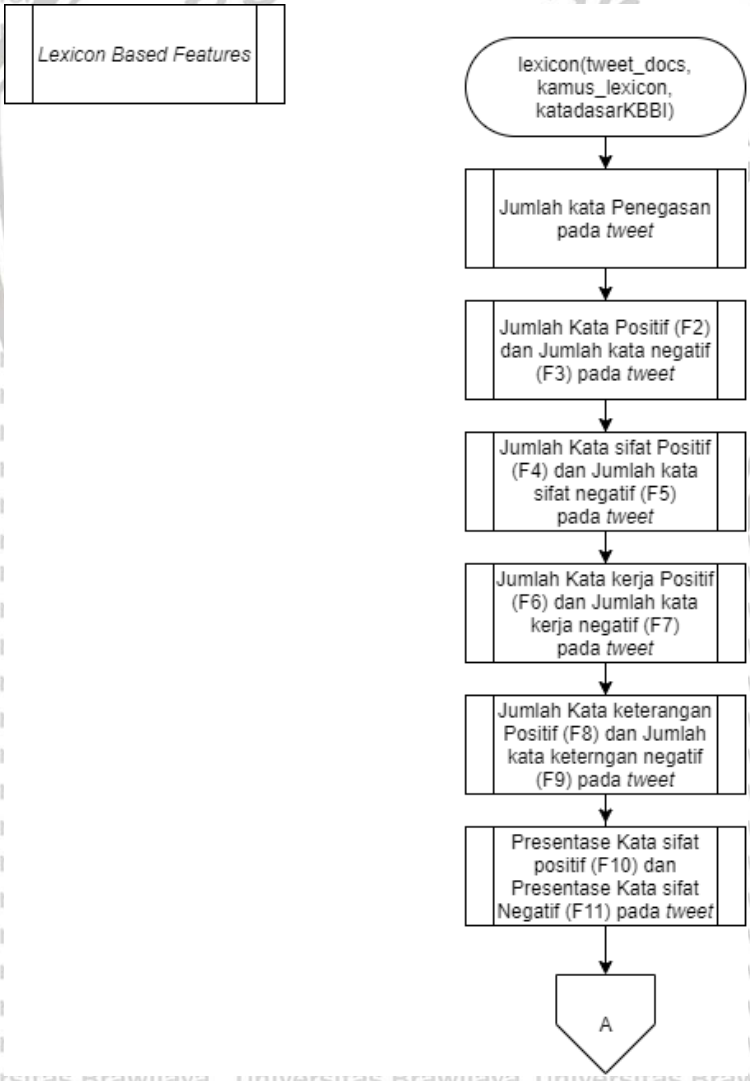
Proses terakhir dari pembobotan kata, adalah perhitungan nilai TF-IDF, perhitungan ini bertujuan agar dapat merepresentasikan bobot di tiap fitur kata, tiap kata yang memiliki nilai bobot tinggi merupakan kata yang jarang atau tidak dapat ditemukan pada dokumen lain, sebaliknya tiap kata yang memiliki bobot rendah berarti sering ditemukan pada dokumen lain, apabila kata tersebut terdapat pada tiap dokumen, maka dapat dipastikan bobot TF-IDF kata tersebut bernilai nol. Pada tahap ini digunakan masukan berupa IDF dan Wtf untuk dapat mengimplementasikan rumus pada Persamaan 2.3. *Output* pada tahap ini yaitu nilai bobot dari setiap kata pada dokumen data latih. Alur perhitungan TF-IDF dapat dilihat dalam Gambar 4.12.




**Gambar 4.12 Diagram Alir Perhitungan TF-IDF**

## 4.4 *Lexicon Based Feature*

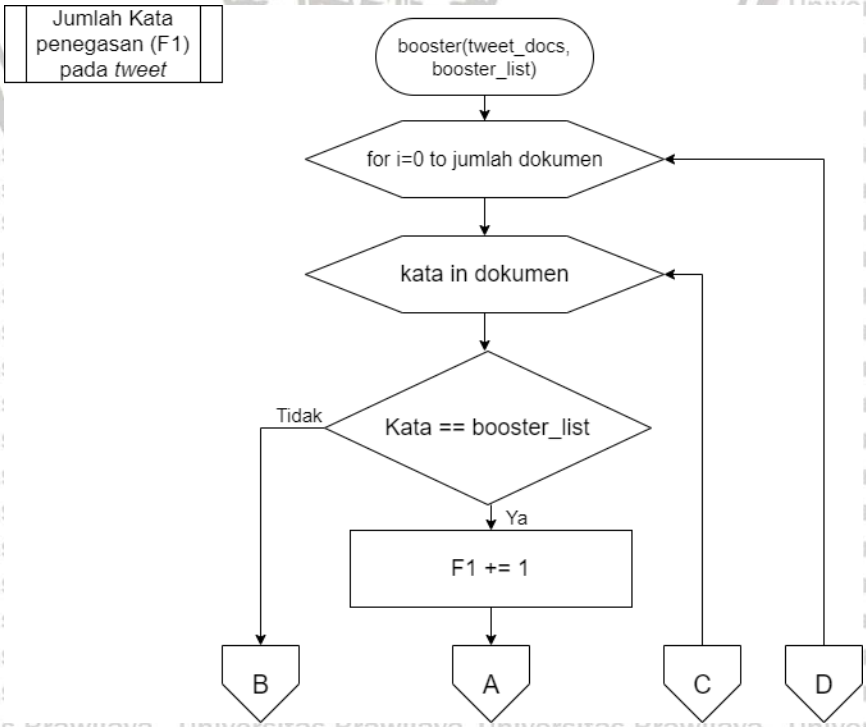
Setelah melakukan pembobotan kata menggunakan TF-IDF, tahap selanjutnya yaitu melakukan pembobotan kata menggunakan *Lexicon Based Feature.* Terdapat 15 fitur yang telah dijabarkan pada tabel 2.1 untuk digunakan untuk menghitung bobot pada dokumen. Tahap awal dari proses penentuan bobot adalah menghitung jumlah dari kata penegasan (F1), setelah itu menghitung jumlah kata positif (F2) dan kata negatif (F3), lalu menghitung jumlah kata positif pada kata sifat (F4) dan kata negatif pada kata sifat (F5), dilanjutkan dengan menghitung jumlah kata positif terhadap kata kerja (F6) dan kata negatif pada kata kerja (F7), lalu dilanjut dengan perhitungan jumlah kata positif pada kata keterangan (F8) dan kata negatif pada kata keterangan (F9), selanjutnya dilakukan perhitungan persentase kata positif pada kata sifat (F10) dan persentase kata negatif pada kata sifat (F11), kemudian persentase kata positif pada kata kerja (F13), tahap terakhir yaitu persentase kata positif pada kata keterangan (F14) dan kata negatif pada kata keterangan (F15). Diagram alir dari *Lexicon Based Feature* telah disajikan pada Gambar 4.13.

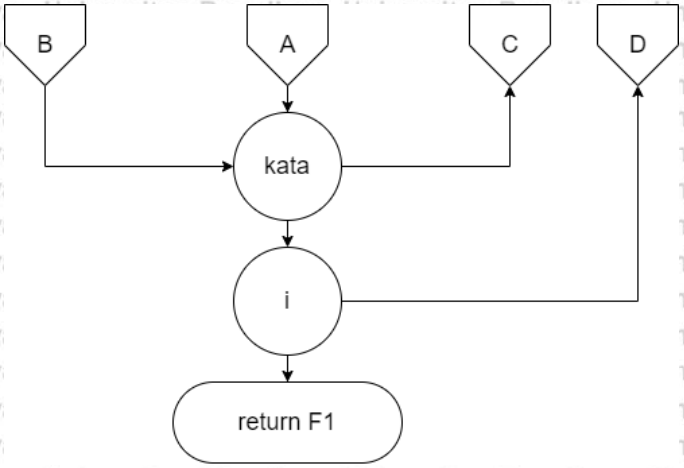



**Gambar 4.13 Diagram Alir *Lexicon Based Feature***

### 4.4.1 Jumlah Kata Penegasan (F1)

Tahap awal perhitungan bobot *lexicon* adalah perhitungan kata penegasan atau *booster words.* Proses perhitungan dilakukan dengan cara mencari setiap kata pada tiap dokumen pada kamus daftar kata penegasan. Masukan dari proses ini adalah dokumen *tweet* dan kamus daftar kata penegasan. *Output* dari proses ini berupa bobot dari F1 pada tiap fitur di setiap dokumen. Diagram alir jumlah kata penegasan telah diuraikan pada Gambar 4.14.

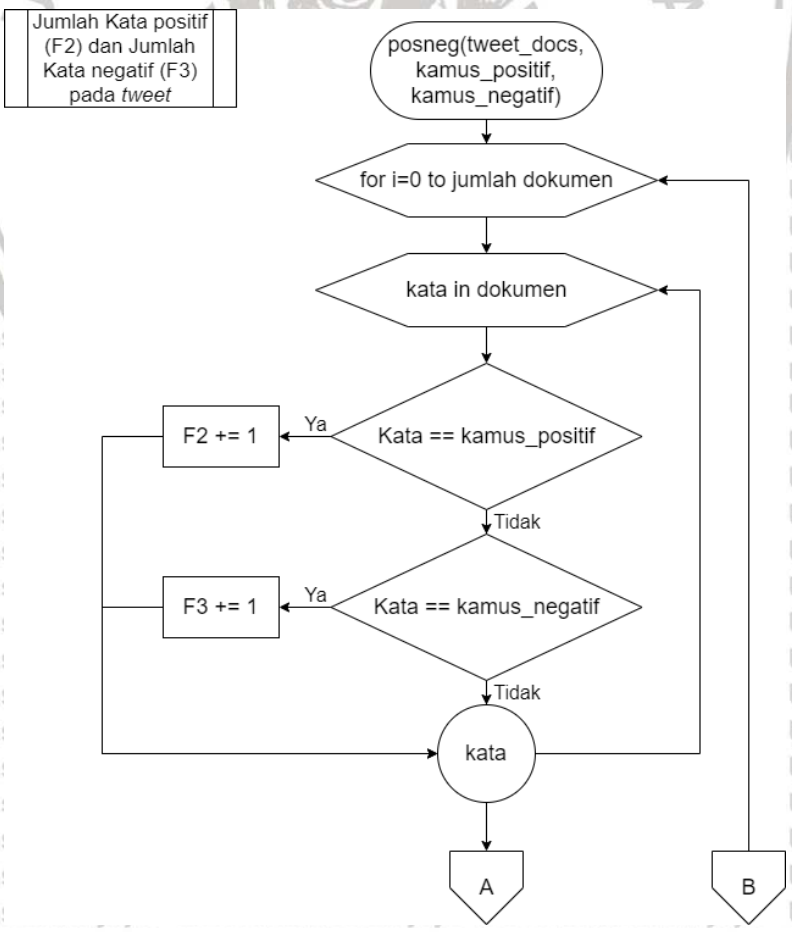


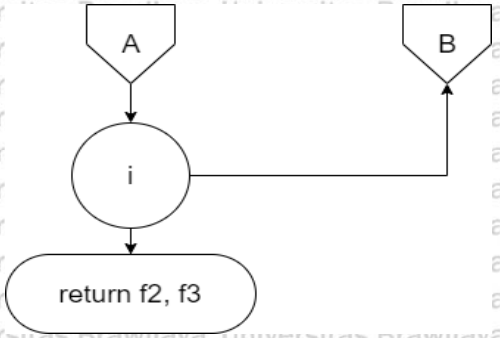



**Gambar 4.14 Diagram Alir F1**

### 4.4.2 Jumlah Kata Positif (F2) dan Jumlah Kata Negatif (F3)

Tahap selanjutnya adalah perhitungan terhadap jumlah kata positif (F1) dan jumlah kata negatif (F2). Proses perhitungan dilakukan dengan mencari tiap kata pada tiap dokumen pada kamus kata positif atau kata negatif. Masukan dari proses ini adalah dokumen *tweet*, kamus daftar kata positif dan kamus daftar kata negatif. *Output* dari proses ini berupa bobot dari F2 dan F3 pada tiap fitur di setiap dokumen. Diagram alir jumlah kata positif dan jumlah kata negatif dapat dilihat dalam Gambar 4.15.

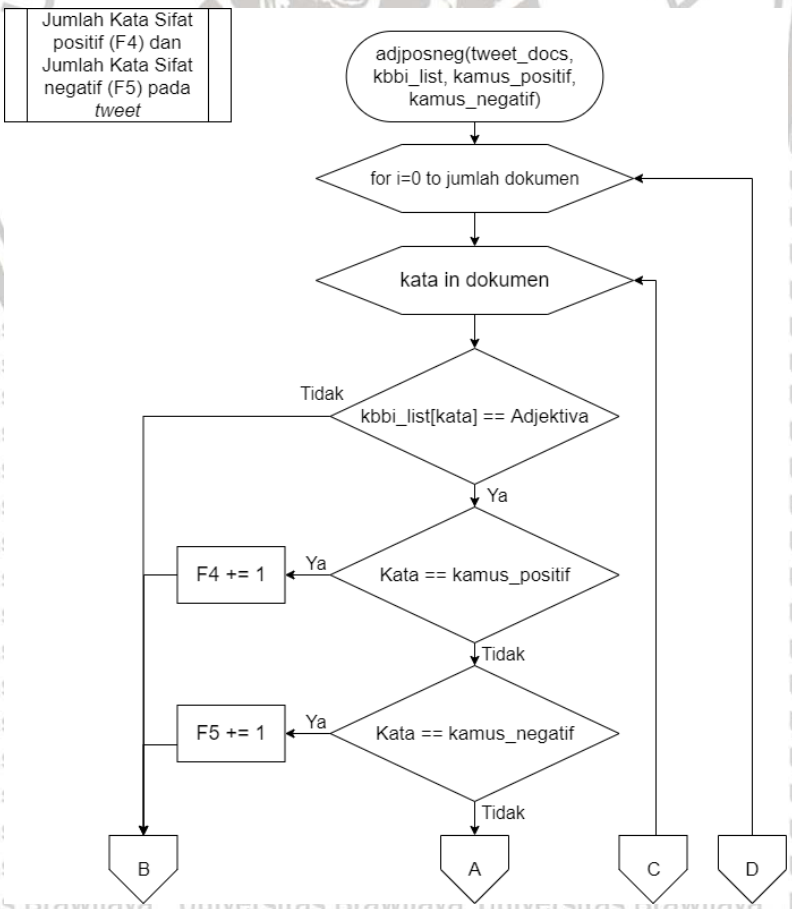


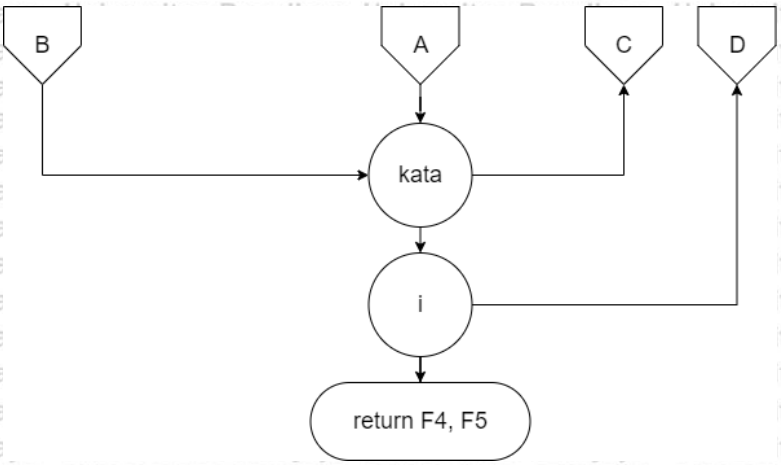



**Gambar 4.15 Diagram Alir F2 dan F3**

### 4.4.3 Jumlah Kata Sifat Positif (F4) dan Jumlah Kata Sifat Negatif (F5)

Tahap selanjutnya adalah perhitungan terhadap jumlah kata positif pada kata sifat (F4) dan jumlah kata negatif pada kata sifat (F5). Proses perhitungan dilakukan dengan mencari jenis kata dari tiap kata pada tiap dokumen pada Kamus Besar Bahasa Indonesia (KBBI) setelah itu mencari sentimen dari kata tersebut pada kamus daftar kata positif atau negatif. Masukan dari proses ini adalah dokumen *tweet*, KBBI, kamus daftar kata positif dan kamus daftar kata negatif. *Output* dari proses ini berupa bobot dari F4 dan F5 pada tiap fitur di setiap dokumen. Diagram alir jumlah kata sifat positif dan jumlah kata sifat negatif dapat dilihat dalam Gambar 4.16.

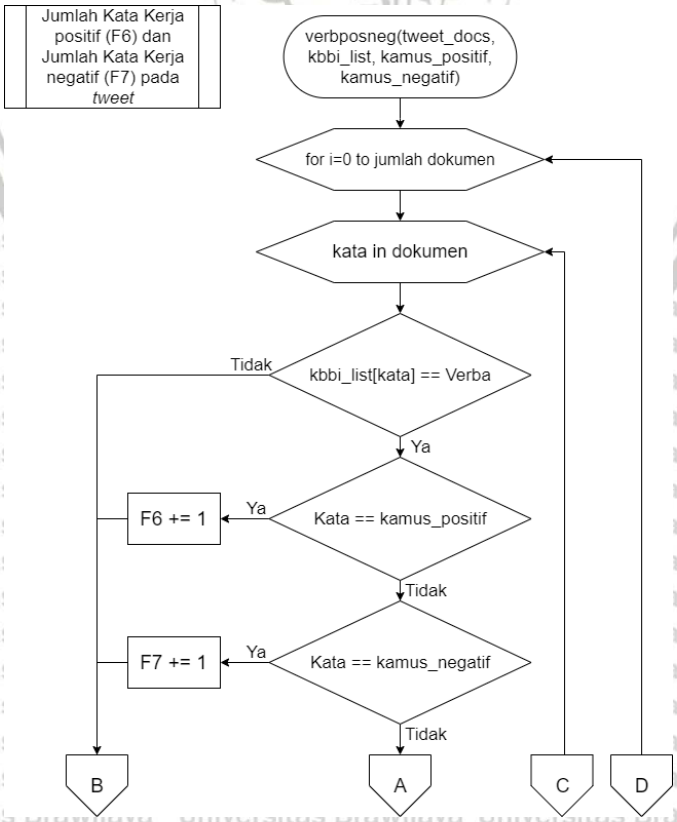


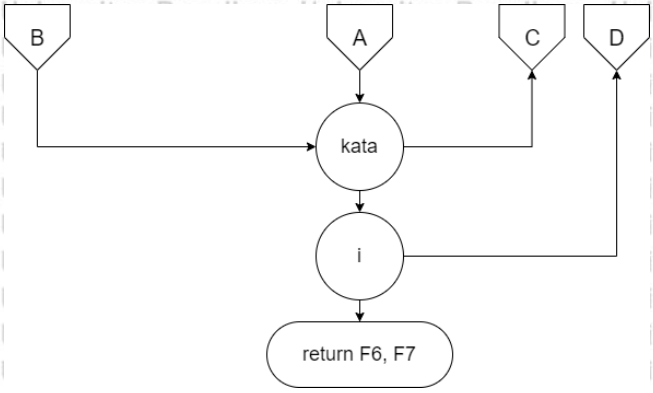



**Gambar 4.16 Diagram Alir F4 dan F5**

### 4.4.4 Jumlah Kata Kerja Positif (F6) dan Jumlah Kata Kerja Negatif (F7)

Tahap selanjutnya adalah perhitungan terhadap jumlah kata positif pada kata kerja (F6) dan jumlah kata negatif pada kata kerja (F7). Proses perhitungan dilakukan dengan mencari jenis kata dari tiap kata pada tiap dokumen pada KBBI setelah itu mencari sentimen dari kata tersebut pada kamus daftar kata positif atau negatif. Masukan dari proses ini adalah dokumen *tweet*, KBBI, kamus daftar kata positif dan kamus daftar kata negatif. *Output* dari proses ini berupa bobot dari F6 dan F7 pada tiap fitur di setiap dokumen. Diagram alir jumlah kata kerja positif dan jumlah kata kerja negatif dapat dilihat dalam Gambar 4.17.

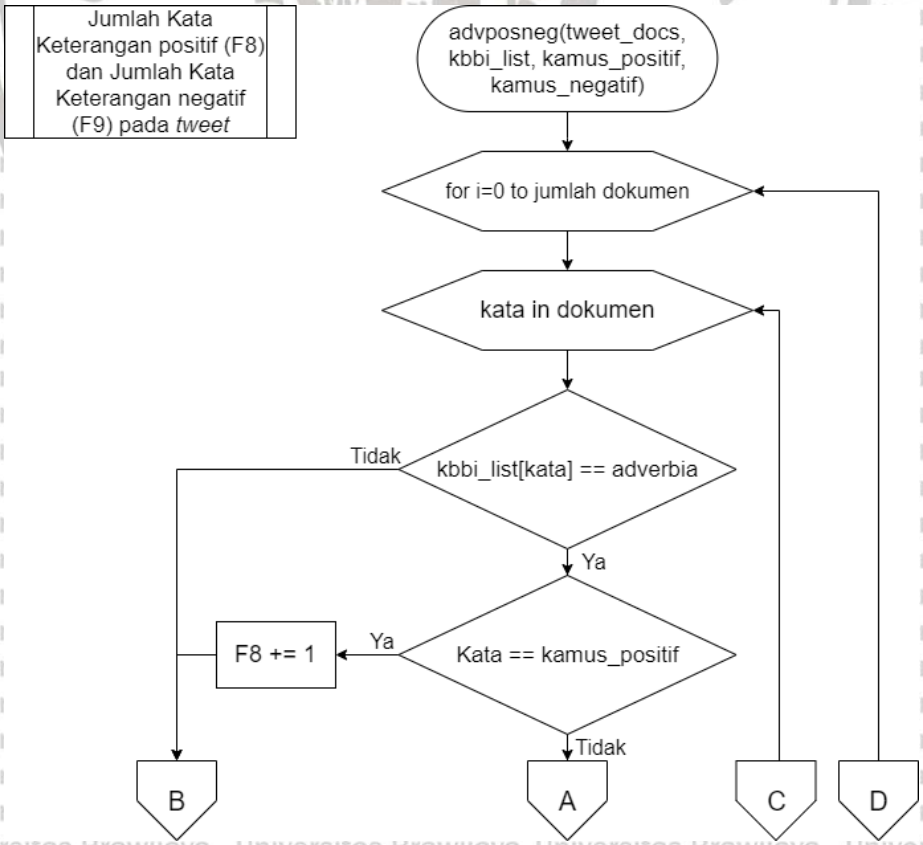


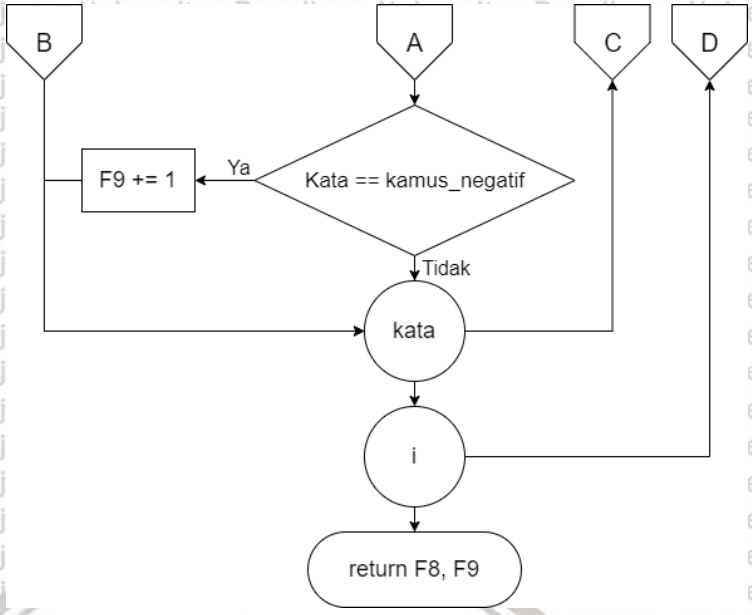



**Gambar 4.17 Diagram Alir F6 dan F7**

### 4.4.5 Jumlah Kata Keterangan Positif (F8) dan Jumlah Kata Keterangan Negatif (F9)

Tahap selanjutnya adalah perhitungan terhadap jumlah kata positif pada kata keterangan (F8) dan jumlah kata negatif pada kata keterangan (F9). Proses perhitungan dilakukan dengan mencari jenis kata dari tiap kata pada tiap dokumen pada KBBI setelah itu mencari sentimen dari kata tersebut pada kamus daftar kata positif atau negatif. Masukan dari proses ini adalah dokumen *tweet*, KBBI, kamus daftar kata positif dan kamus daftar kata negatif. *Output* dari proses ini berupa bobot dari F8 dan F9 pada tiap fitur di setiap dokumen. Diagram alir jumlah kata kerja positif dan jumlah kata kerja negatif dapat dilihat dalam Gambar 4.18.

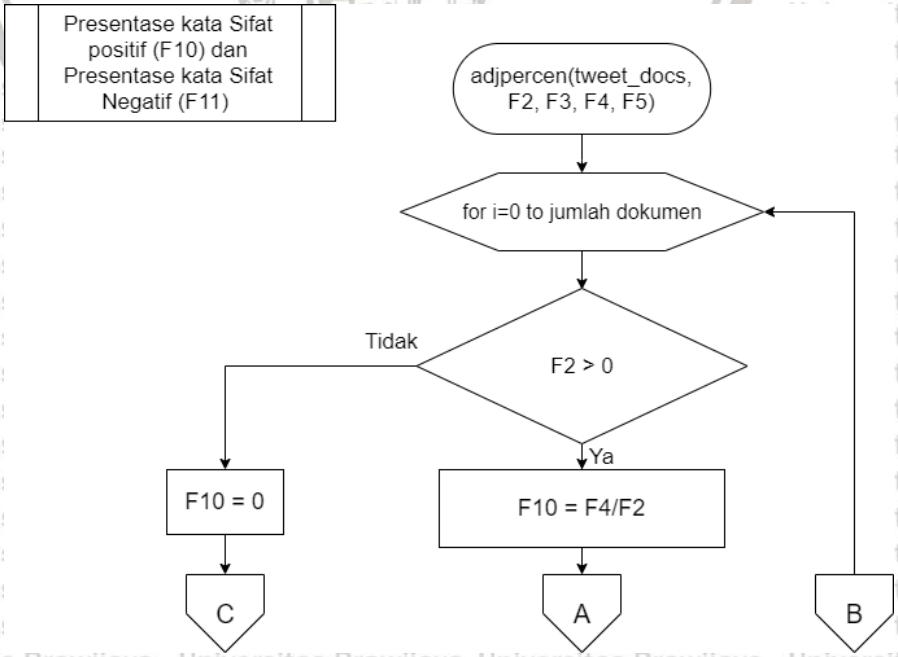


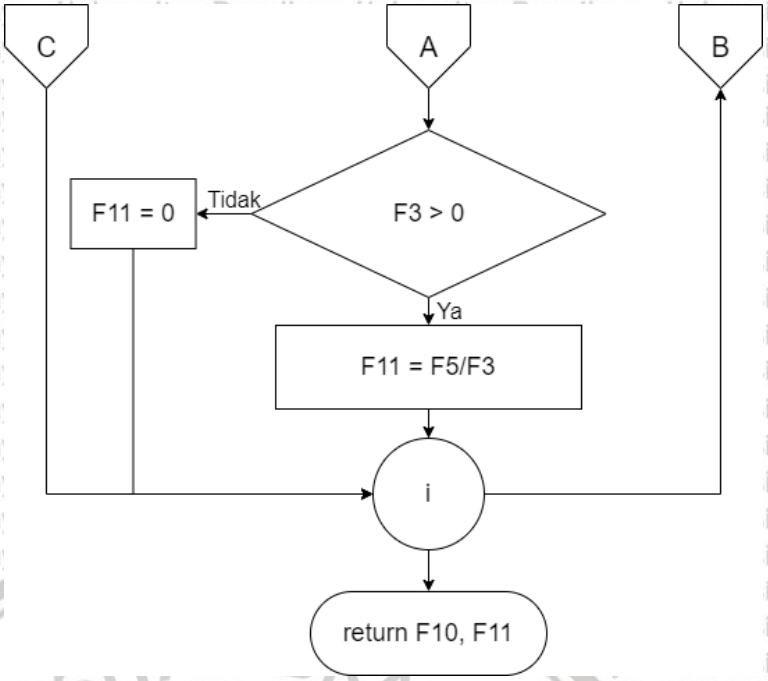



**Gambar 4.18 Diagram Alir F8 dan F9**

### 4.4.6 Persentase Kata Sifat Positif (F10) dan Persentase Kata Sifat Negatif (F11)

Tahap selanjutnya adalah perhitungan terhadap persentase kata positif pada kata sifat (F10) dan persentase kata negatif pada kata sifat (F11). Proses ini dilakukan dengan cara membagi nilai dari kata sifat positif dengan jumlah kata positif dan nilai dari kata sifat negatif dengan jumlah kata negatif pada tiap dokumen. Masukan dari proses ini adalah dokumen *tweet*, nilai dari F2, nilai dari F3, nilai dari F4, dan nilai dari F5. *Output* dari tahapan ini adalah bobot dari F10 dan F11. Diagram alir persentase kata sifat positif dan persentase kata sifat negatif dapat dilihat pada Gambar 4.19.

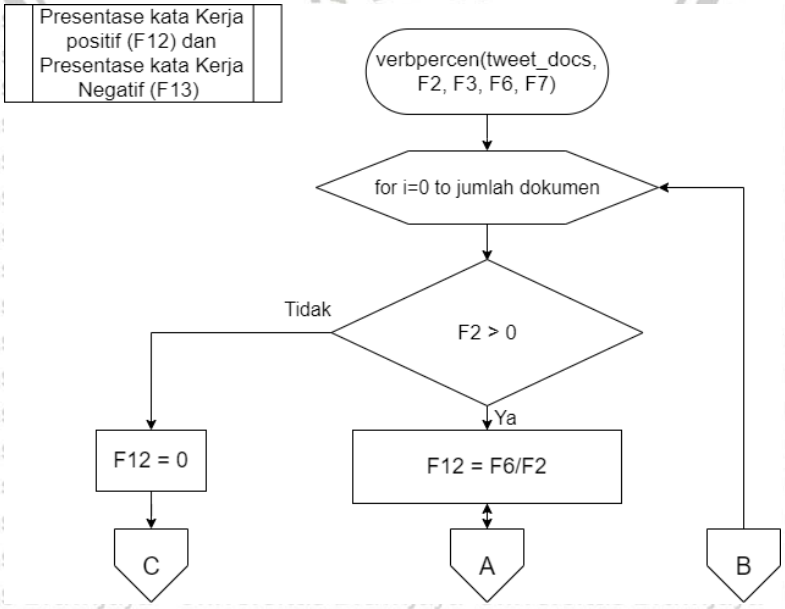


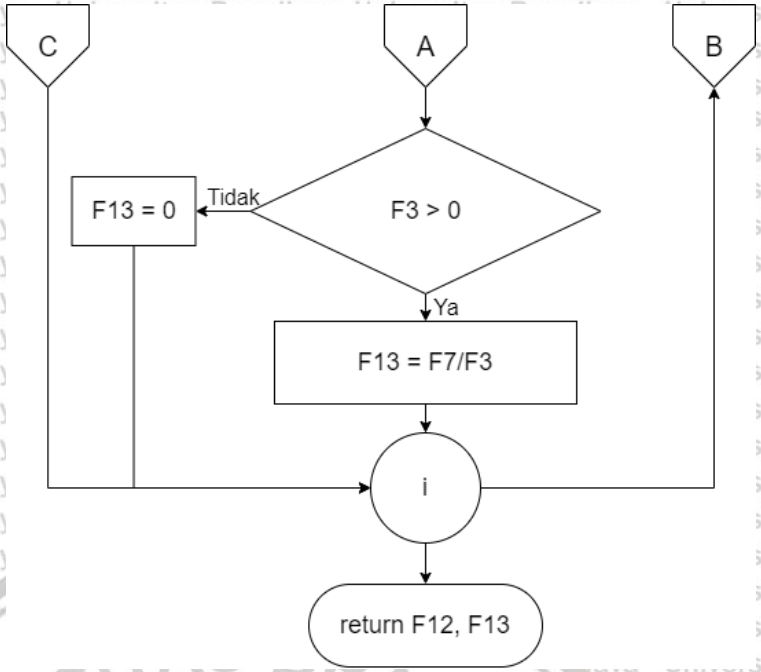



**Gambar 4.19 Diagram Alir F10 dan F11**

### 4.4.7 Persentase Kata Kerja Positif (F12) dan Persentase Kata Kerja Negatif (F13)

Tahap selanjutnya adalah perhitungan terhadap persentase kata positif pada kata kerja (F12) dan persentase kata negatif pada kata kerja (F13). Proses ini dilakukan dengan cara membagi nilai dari kata kerja positif dengan jumlah kata positif dan nilai dari kata kerja negatif dengan jumlah kata negatif pada tiap dokumen. Masukan dari proses ini adalah dokumen *tweet*, nilai dari F2, nilai dari F3, nilai dari F6, dan nilai dari F7. *Output* dari tahapan ini adalah bobot dari F12 dan F13. Diagram alir persentase kata kerja positif dan persentase kata kerja negatif dapat dilihat pada Gambar 4.20.

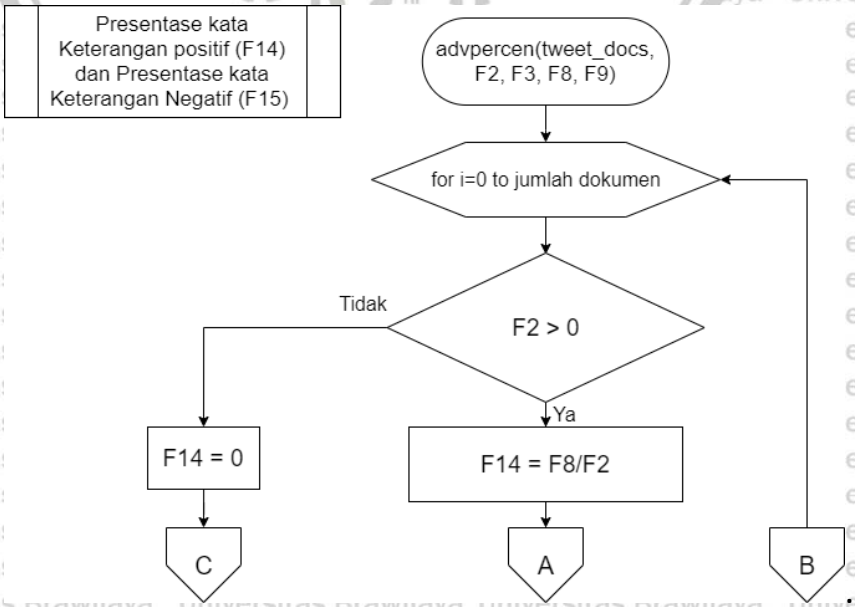


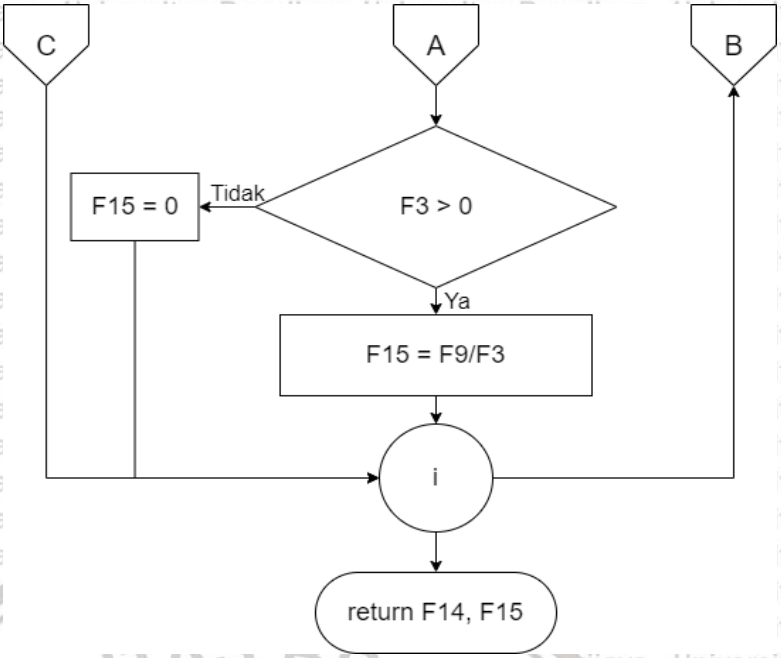



**Gambar 4.20 Diagram Alir F12 dan F13**

### 4.4.8 Persentase Kata Keterangan Positif (F14) dan Persentase Kata Keterangan Negatif (F15)

Tahap selanjutnya adalah perhitungan terhadap persentase kata positif pada kata keterangan (F14) dan persentase kata negatif pada kata keterangan (F15). Proses ini dilakukan dengan cara membagi nilai dari kata keterangan positif dengan jumlah kata positif dan nilai dari kata keterangan negatif dengan jumlah kata negatif pada tiap dokumen. Masukan dari proses ini adalah dokumen *tweet*, nilai dari F2, nilai dari F3, nilai dari F8, dan nilai dari F9. *Output* dari tahapan ini adalah bobot dari F14 dan F15. Diagram alir persentase kata keterangan positif dan persentase kata keterangan negatif dapat dilihat dalam Gambar 4.21.

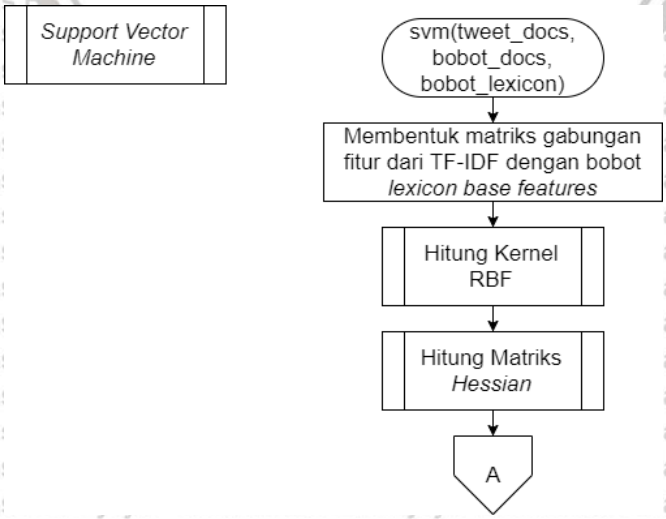


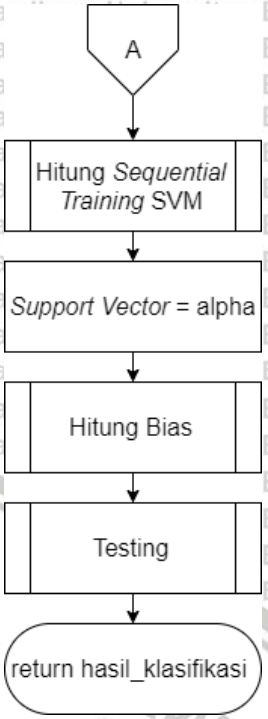



**Gambar 4.21 Diagram Alir F14 dan F15**

## 4.5 Support Vector Machine

Setelah mendapatkan nilai dari bobot TF-IDF dan *Lexicon Based Feature*, tahap selanjutnya adalah penerapan algoritme Support Vector Machine. Sebelum memulai tahapan ini, diperlukan inisialisasi dari nilai *sigma*, *lambda*, *gamma*, *epsilon*, konstanta, dan jumlah iterasi maksimum untuk melatih data latih. Masukan dari proses ini adalah dokumen *tweet,* bobot TF-IDF dan *Lexicon Based Feature* yang kemudian digabung menjadi bobot dari fitur pada dokumen. *Output* dari tahapan ini berupa kelas dari dokumen data uji. Tahapan dari penerapan algoritme SVM adalah menggabungkan bobot TF-IDF dan *Lexicon Based Feature*, perhitungan kernel, perhitungan matriks Hessian, *Sequential Training* terhadap data latih, inisialisasi variabel Support Vector perhitungan nilai bias, dan *testing*. Diagram alir dari Support Vector Machine dapat dilihat pada Gambar 4.22.

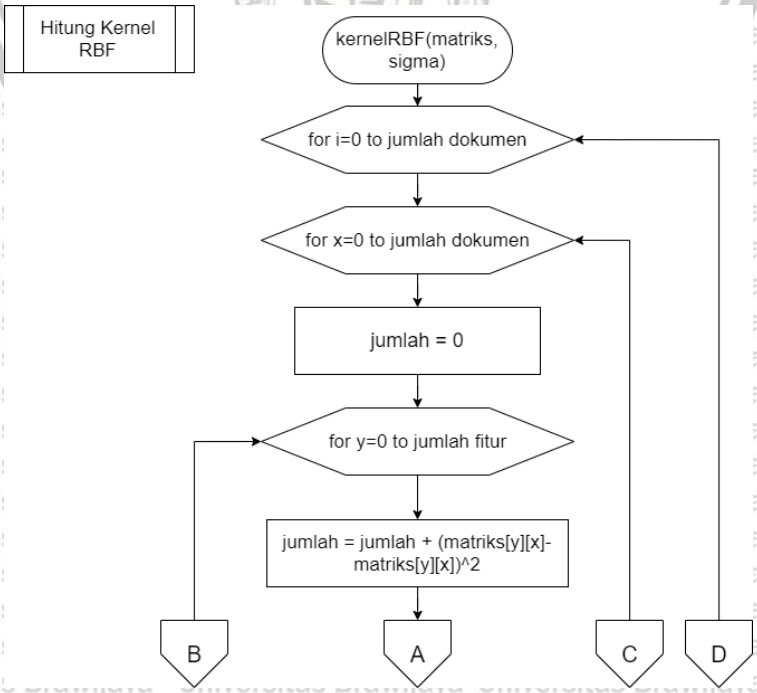


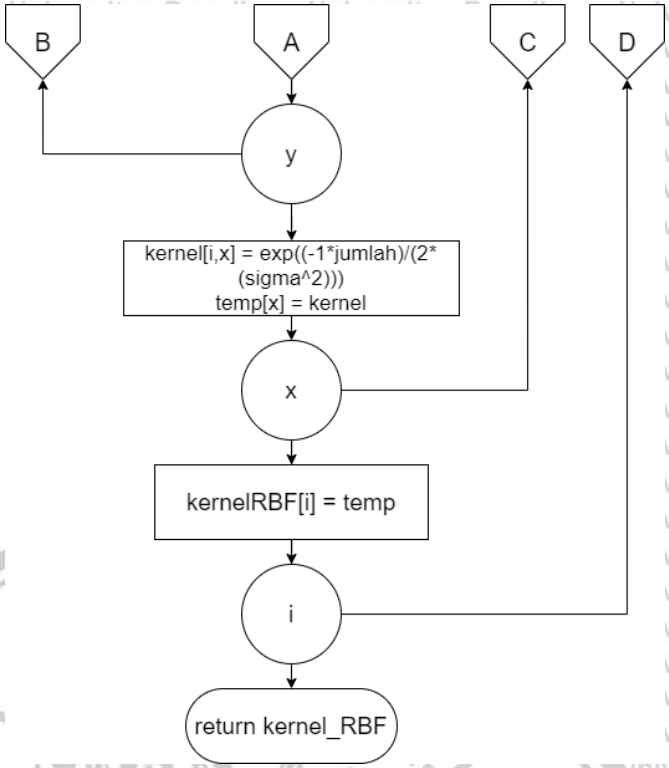



**Gambar 4.22 Diagram Alir Support Vector Machine**

### 4.5.1 Hitung Kernel RBF

Tahap pertama dari penerapan algoritme Support Vector Machine adalah perhitungan Kernel RBF. Proses perhitungan Kernel RBF dimulai dengan menginisialisasi nilai dari sigma. Masukan pada proses ini adalah matriks gabungan dari bobot TF-IDF dan *Lexicon Based Feature*. Output dari tahapan ini adalah nilai dari Kernel RBF pada tiap dokumen. Alur perhitungan Kernel RBF dapat dilihat pada Gambar 4.23.

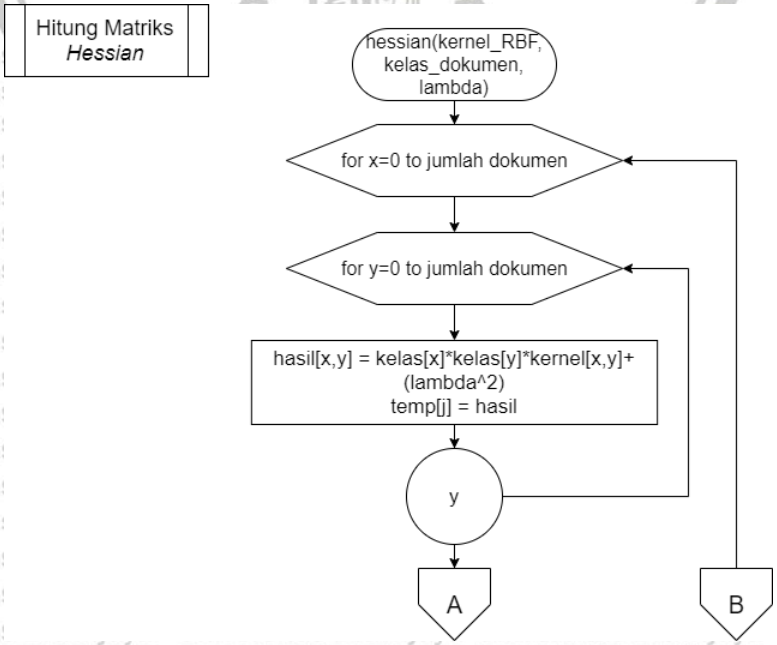


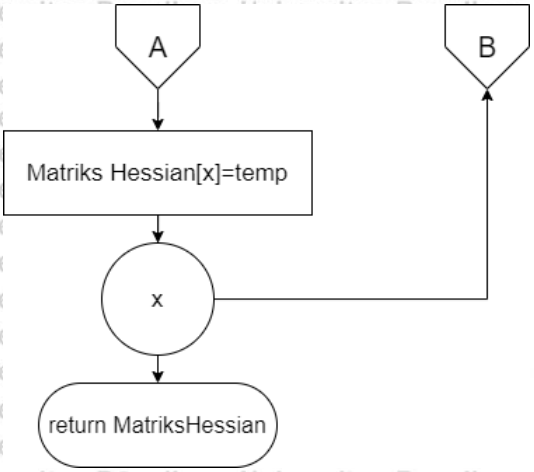



**Gambar 4.23 Diagram Alir Kernel RBF**

### 4.5.2 Matriks Hessian

Tahap selanjutnya adalah perhitungan matriks Hessian dengan menggunakan rumus pada Persamaan 2.5. Sebelum melakukan perhitungan diperlukan inisialisasi nilai *lambda*. Masukan pada proses ini adalah nilai dari kernel_RBF, serta kelas dari dokumen. *Output* dari tahapan ini adalah nilai dari matriks Hessian pada tiap dokumen. Alur dari perhitungan matriks Hessian dapat dilihat pada Gambar 4.24.

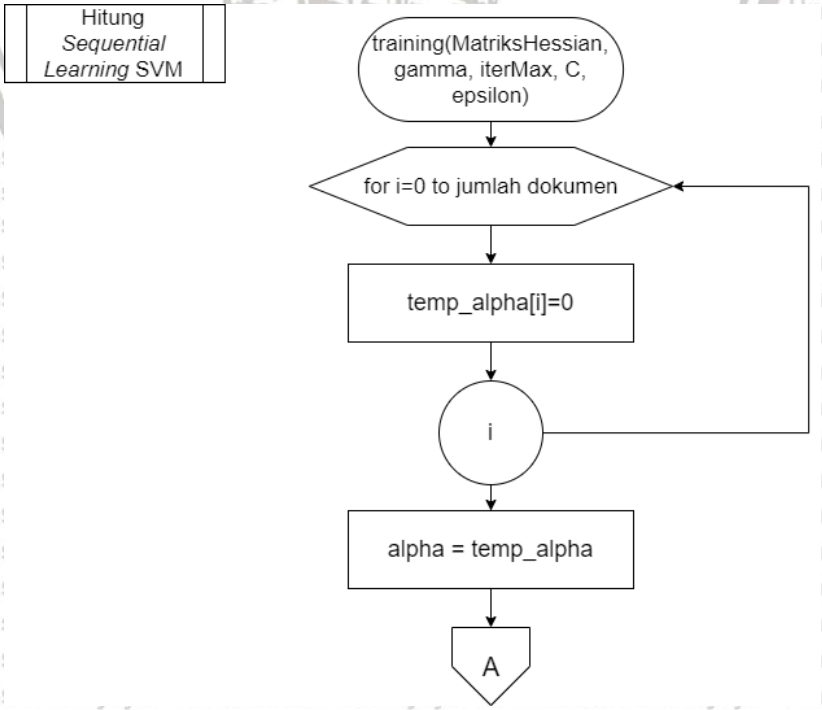


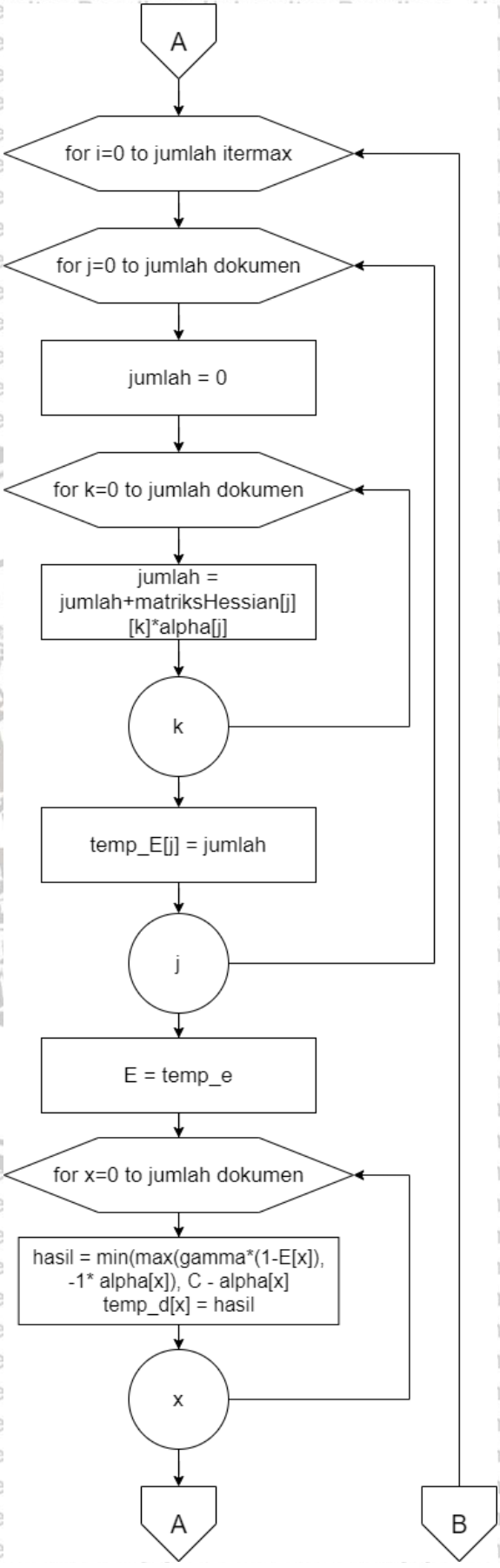


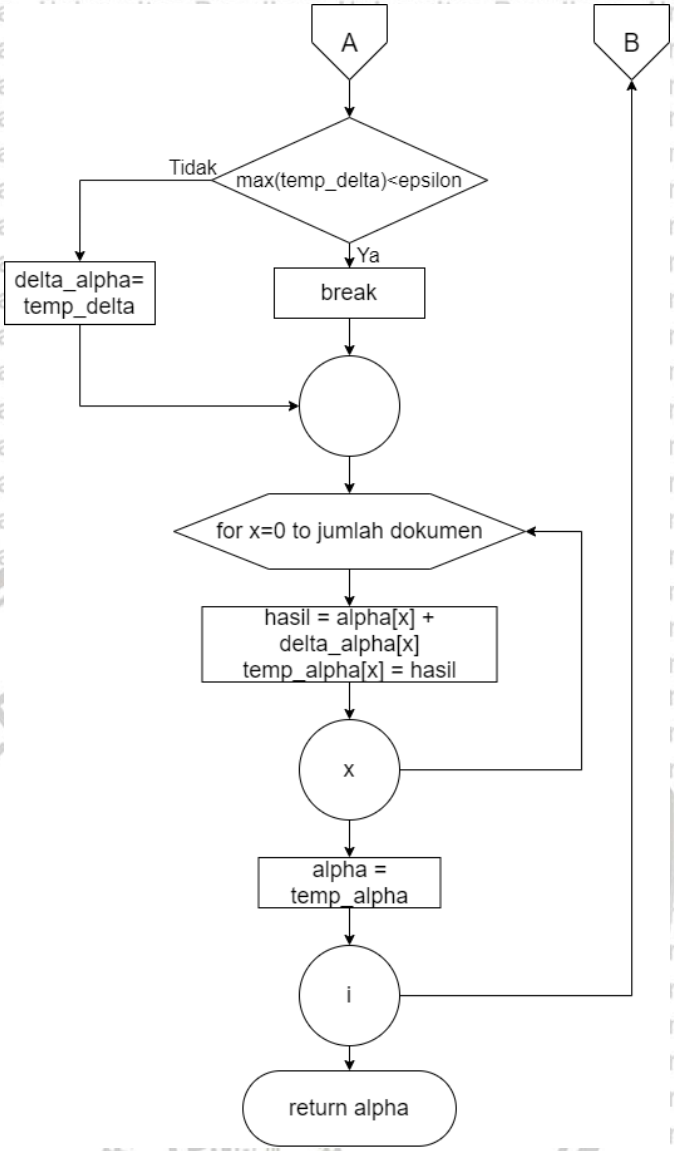

**Gambar 4.24 Diagram Alir Matriks Hessian**

### *4.5.3 Sequential Training SVM*

Tahap selanjutnya yaitu melakukan pelatihan terhadap data uji menggunakan *Sequential Training*. Pada tahap ini diperlukan adanya inisialisasi nilai konstanta C, jumlah itermax, epsilon, alpha dan gamma. Masukan pada proses ini adalah matriks Hessian. Pada proses ini dilakukan perulangan untuk mendapatkan nilai *error rate* dengan menggunakan rumus pada Persamaan 2.6, delta alpha dengan menggunakan rumus pada Persamaan 2.7, dan alpha dengan menggunakan rumus pada Persamaan 2.8. Perulangan berhenti apabila nilai perulangan sudah mencapai nilai iter max yang telah di inisialisasi atau apabila nilai dari delta alpha lebih kecil dari nilai epsilon yang telah di inisialisasi. *Output* dari proses ini adalah nilai alpha yang digunakan sebagai support vector di tahap selanjutnya. Diagram alir *Sequential Training* dapat dilihat pada Gambar 4.25.



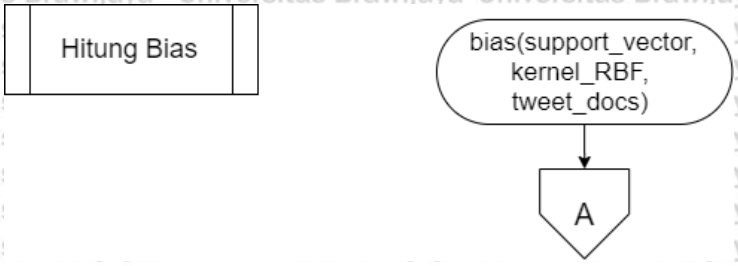

A
for i=0 to jumlah itermax
for j=0 to jumlah dokumen
jumlah = 0
for k=0 to jumlah dokumen
jumlah = jumlah+matriksHessian[j][k]*alpha[j]
k
temp_E[j] = jumlah
j
E = temp_e
for x=0 to jumlah dokumen
hasil = min(max(gamma*(1-E[x]), -1* alpha[x]), C - alpha[x]
temp_d[x] = hasil
x
A
B

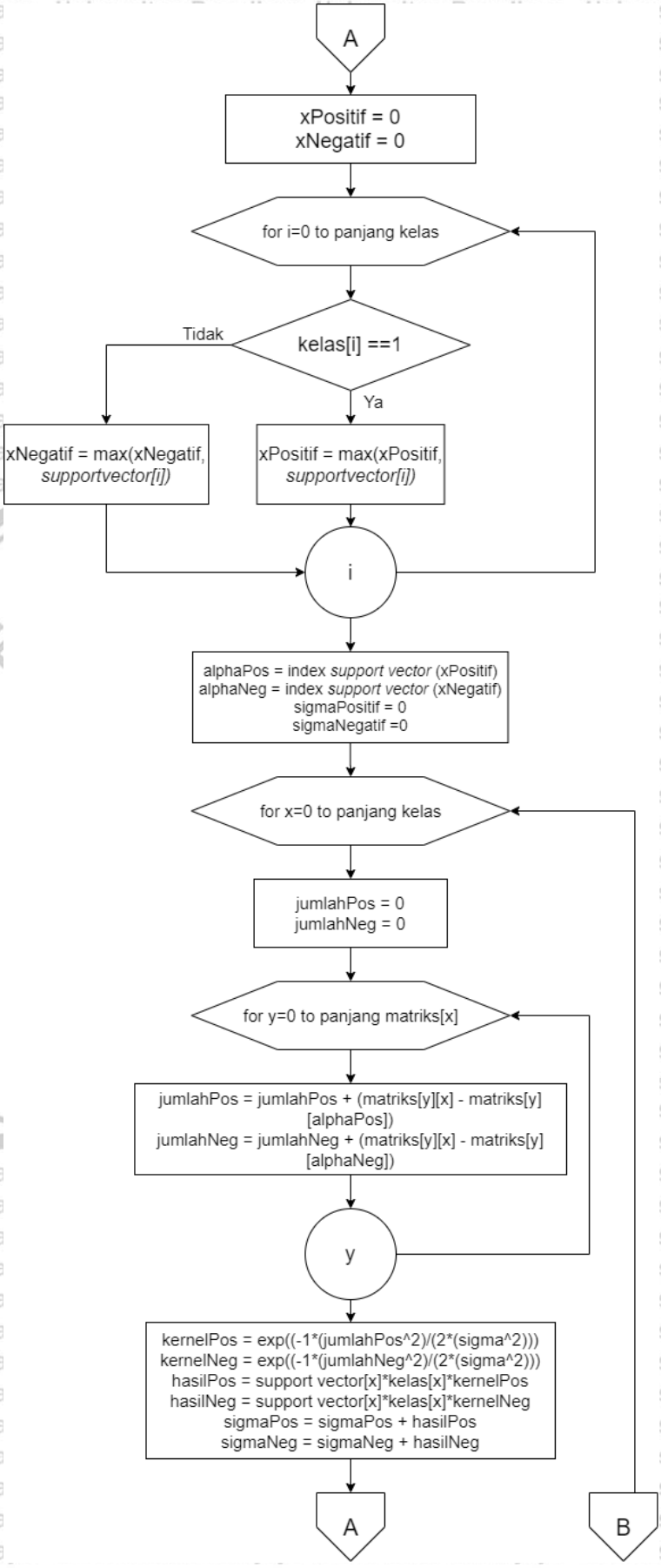


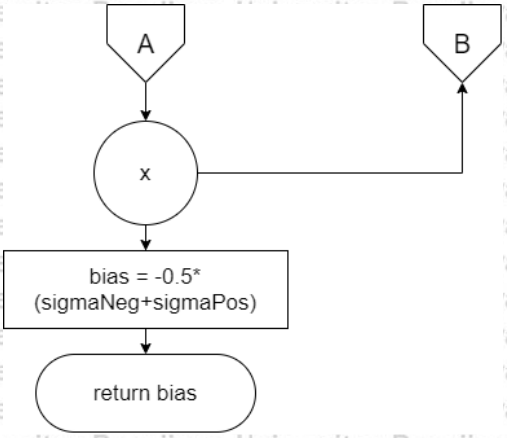

**Gambar 4.25 Diagram Alir Sequential Training SVM**

### 4.5.4 Hitung Bias

Setelah mendapatkan nilai alpha, dilakukan tahap menghitung nilai bias dengan menggunakan rumus pada Persamaan 2.10 dan Kernel RBF dari indeks yang memiliki nilai support vector terbesar dari kelas positif dan kelas negatif. Masukan pada proses ini adalah nilai support vector, Kernel RBF, dan kelas dari dokumen. *Output* dari tahapan ini adalah nilai dari bias. Alur perhitungan nilai bias dapat dilihat pada Gambar 4.26.



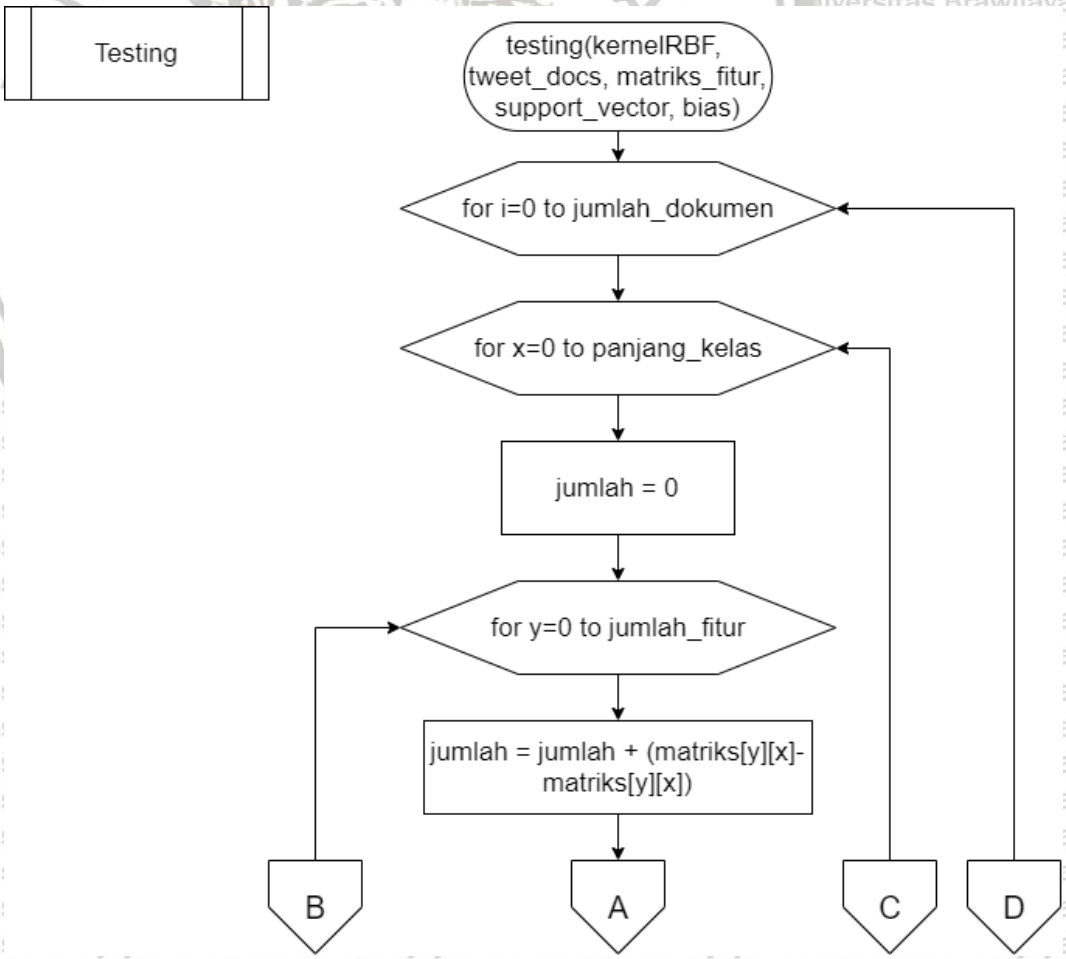

A
xPositif = 0
xNegatif = 0
for i=0 to panjang kelas
Tidak
kelas[i] ==1
Ya
xNegatif = max(xNegatif, supportvector[i])
xPositif = max(xPositif, supportvector[i])
i
alphaPos = index support vector (xPositif)
alphaNeg = index support vector (xNegatif)
sigmaPositif = 0
sigmaNegatif =0
for x=0 to panjang kelas
jumlahPos = 0
jumlahNeg = 0
for y=0 to panjang matriks[x]
jumlahPos = jumlahPos + (matriks[y][x] - matriks[y][alphaPos])
jumlahNeg = jumlahNeg + (matriks[y][x] - matriks[y][alphaNeg])
y
kernelPos = exp((-1*(jumlahPos^2)/(2*(sigma^2)))
kernelNeg = exp((-1*(jumlahNeg^2)/(2*(sigma^2)))
hasilPos = support vector[x]*kelas[x]*kernelPos
hasilNeg = support vector[x]*kelas[x]*kernelNeg
sigmaPos = sigmaPos + hasilPos
sigmaNeg = sigmaNeg + hasilNeg
A
B

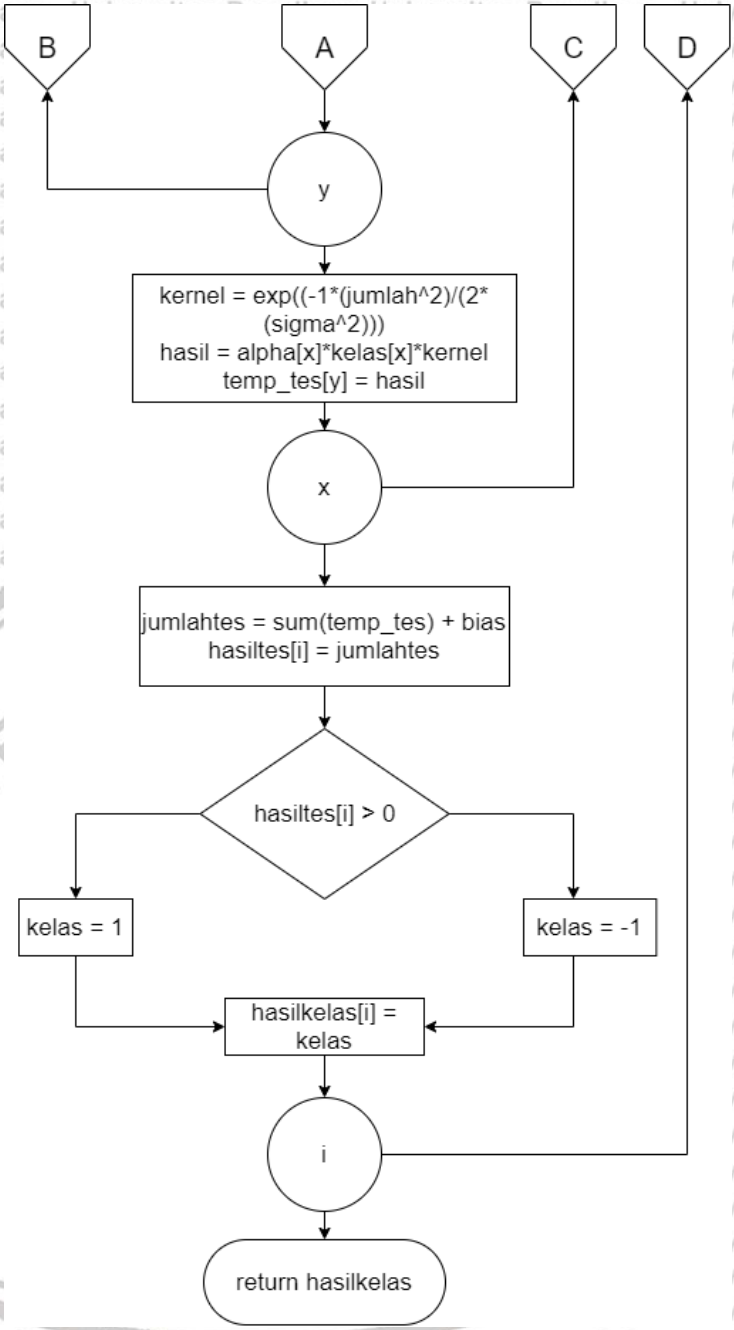



**Gambar 4.26 Diagram Alir Hitung Bias**

### *4.5.5 Testing*

Tahap terakhir dari Support Vector Machine adalah tahap *testing*. Tahap ini adalah tahap untuk mendapatkan kelas dari data uji. Untuk melakukan perhitungan pada data uji, data uji perlu melewati tahap *preprocessing* hingga perhitungan Kernel RBF. Masukan dari proses ini adalah dokumen data uji, kelas data latih, Support Vector, dan bias. *Output* dari tahapan ini adalah hasil dari kelas data uji. Diagram alir proses *testing* telah diuraikan pada Gambar 4.27.






**Gambar 4.27 Diagram Alir Testing**

## 4.6 Perhitungan Manual

Manualisasi dilakukan pada data yang telah diambil yang berjumlah 8 data, dan mempunyai 4 kelas positif dan 4 kelas negatif, dan dibagi menjadi 6 data latih dan 2 data uji. Data latih melewati tahap *preprocessing* hingga tahap Support Vector Machine. Sementara itu, data uji melewati tahap *preprocessing* hingga tahap perhitungan Kernel RBF. Dokumen data latih dan dokumen data uji yang digunakan dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

**Tabel 4.1 Data Latih**

| No | *Tweet* | Kelas |
|---|---|---|
| 1 | @QaillaAsyiqah Semuga allah segera merobek-robek kekuasaanya @jokowi dan jajarannya. | -1 |
| 2 | Pasti bangga dong dgn kinerja Presiden @jokowi yg sdh berhasil dan bikin spt ini. #IndonesiaMaju https://t.co/OGZzBsb7xC | 1 |
| 3 | Aset negara yg najis2 haram jadah pantes negerimu diambang kehancuran @kyai_marufAmin @jokowi @DPR_RI | -1 |
| 4 | Terima pak @jokowi dan @basuki_btp atas sumbangsihnya dalam pembangunan kota Jakarta hingga di nobatkan sebagai Kota Terbaik Dunia. ðŸ™ | 1 |
| 5 | Selamat Hari Pahlawan 2020. Terimakasih pak @jokowi ðŸ™ https://t.co/9jR88Jce28 | 1 |
| 6 | @jokowi @basuki_btp @aniesbaswedan Pak Boss.. gimana rasanya sakit hati kala melihat Pak Anies Baswedan penuh Prestasi. Sedangkan Junjunan anda...?? Apa prestasinya..?? NOL BESAR Apalagi si Ahok yg dulu sering ngomong Tai... | -1 |

**Tabel 4.2 Data Uji**

| | |
|---|---|
| 1 | presiden @Jokowi mengatakan dalam keberagaman perlu terus dilakukan untuk saling toleransi dan solidaritas guna pondasi yg kokoh dalam hidup dalam keberagaman. #indonesiamaju #jokowi https://t.co/VGTxjAFXLE |
| 2 | Nak SD aja paham ini UU sampah. Pak @jokowi otak udah gak bisa mikir ya? atau emang gak ada otak? Presiden gak ada beban ya gini nih.... #MosiTidakPercaya |

### *4.6.1 Cleaning*

Tahap pertama yang dilakukan adalah mengolah data menjadi dokumen yang hanya berisikan teks. Proses ini dilakukan dengan cara menghilangkan *noise* berupa *link,* angka, dan symbol. Tahap *cleaning* telah disajikan dalam Tabel 4.3 dan Tabel 4.4.

**Tabel 4.3 Hasil Cleaning pada Data Latih**

| No | Data Latih | Hasil Cleaning |
|---|---|---|
| 1 | @QaillaAsyiqah Semuga allah segera merobek-robek kekuasaanya @jokowi dan jajarannya. | QaillaAsyiqah Semuga allah segera merobek robek kekuasaanya jokowi dan jajarannya |

| | | |
|---|---|---|
| 2 | Pasti bangga dong dgn kinerja Presiden @jokowi yg sdh berhasil dan bikin spt ini. #IndonesiaMaju https://t.co/OGZzBsb7xC | pasti bangga dong dgn kinerja Presiden jokowi yg sdh berhasil dan bikin spt ini IndonesiaMaju |
| 3 | Aset negara yg najis2 haram jadah pantes negerimu diambang kehancuran @kyai_marufAmin @jokowi @DPR_RI | Aset negara yg najis haram jadah pantes negerimu diambang kehancuran kyai marufAmin jokowi DPR RI |
| 4 | Terima pak @jokowi dan @basuki_btp atas sumbangsihnya dalam pembangunan kota Jakarta hingga di nobatkan sebagai Kota Terbaik Dunia. ðŸ™ | Terima pak jokowi dan basuki btp atas sumbangsihnya dalam pembangunan kota Jakarta hingga di nobatkan sebagai Kota Terbaik Dunia |
| 5 | Selamat Hari Pahlawan 2020. Terimakasih pak @jokowi ðŸ™ https://t.co/9jR88Jce28 | Selamat Hari Pahlawan Terimakasih pak jokowi |
| 6 | @jokowi @basuki_btp @aniesbaswedan Pak Boss.. gimana rasanya sakit hati kala melihat Pak Anies Baswedan penuh Prestasi. Sedangkan Junjunan anda...?? Apa prestasinya..?? NOL BESAR Apalagi si Ahok yg dulu sering ngomong Tai... | jokowi basuki btp aniesbaswedan Pak Boss gimana rasanya sakit hati kala melihat Pak Anies Baswedan penuh Prestasi Sedangkan Junjunan anda Apa prestasinya NOL BESAR Apalagi si Ahok yg dulu sering ngomong Tai |

**Tabel 4.4 Hasil Cleaning pada Data Uji**

| No | Data Uji | Hasil Cleaning |
|---|---|---|
| 1 | presiden @Jokowi mengatakan dalam keberagaman perlu terus dilakukan untuk saling toleransi dan solidaritas guna pondasi yg kokoh dalam hidup dalam keberagaman. #indonesiamaju #jokowi https://t.co/VGTxjAFXLE | presiden Jokowi mengatakan dalam keberagaman perlu terus dilakukan untuk saling toleransi dan solidaritas guna pondasi yg kokoh dalam hidup dalam keberagaman indonesiamaju jokowi |
| 2 | Nak SD aja paham ini UU sampah. Pak @jokowi otak udah gak bisa mikir ya? atau emang gak ada otak? Presiden gak ada beban ya gini nih.... #MosiTidakPercaya | Nak SD aja paham ini UU sampah Pak jokowi otak udah gak bisa mikir ya atau emang gak ada otak Presiden gak ada beban ya gini nih MosiTidakPercaya |

### *4.6.2 Case Folding*

Tahap kedua yang dilakukan *case folding,* yaitu mengubah semua kata pada dokumen menjadi huruf kecil. Tahap *case folding* telah disajikan dalam Tabel 4.5 dan Tabel 4.6.

**Tabel 4.5 Hasil Case Folding pada Data Latih**

| No | Hasil *Cleaning* | Hasil *Case Folding* |
|---|---|---|
| 1 | QaillaAsyiqah Semuga allah segera merobek robek kekuasaanya jokowi dan jajarannya | qaillaasyiqah semuga allah segera merobek robek kekuasaanya jokowi dan jajarannya |
| 2 | pasti bangga dong dgn kinerja Presiden jokowi yg sdh berhasil dan bikin spt ini IndonesiaMaju | pasti bangga dong dgn kinerja presiden jokowi yg sdh berhasil dan bikin spt ini indonesiamaju |
| 3 | Aset negara yg najis haram jadah pantes negerimu diambang kehancuran kyai marufAmin jokowi DPR RI | aset negara yg najis haram jadah pantes negerimu diambang kehancuran kyai marufamin jokowi dpr ri |
| 4 | Terima pak jokowi dan basuki btp atas sumbangsihnya dalam pembangunan kota Jakarta hingga di nobatkan sebagai Kota Terbaik Dunia | terima pak jokowi dan basuki btp atas sumbangsihnya dalam pembangunan kota jakarta hingga di nobatkan sebagai kota terbaik dunia |
| 5 | Selamat Hari Pahlawan Terimakasih pak jokowi | selamat hari pahlawan terimakasih pak jokowi |
| 6 | jokowi basuki btp aniesbaswedan Pak Boss gimana rasanya sakit hati kala melihat Pak Anies Baswedan penuh Prestasi Sedangkan Junjunan anda Apa prestasinya NOL BESAR Apalagi si Ahok yg dulu sering ngomong Tai | jokowi basuki btp aniesbaswedan pak boss gimana rasanya sakit hati kala melihat pak anies baswedan penuh prestasi sedangkan junjunan anda apa prestasinya nol besar apalagi si ahok yg dulu sering ngomong tai |

**Tabel 4.6 Hasil Case Folding pada Data Uji**

| No | Hasil *Cleaning* | Hasil *Case Folding* |
|---|---|---|
| 1 | presiden Jokowi mengatakan dalam keberagaman perlu terus dilakukan untuk saling toleransi dan solidaritas guna pondasi yg kokoh dalam hidup | presiden jokowi mengatakan dalam keberagaman perlu terus dilakukan untuk saling toleransi dan solidaritas guna pondasi yg kokoh |

| | | |
|---|---|---|
| | dalam keberagaman indonesiamaju jokowi | dalam hidup dalam keberagaman indonesiamaju jokowi |
| 2 | Nak SD aja paham ini UU sampah Pak jokowi otak udah gak bisa mikir ya atau emang gak ada otak Presiden gak ada beban ya gini nih MosiTidakPercaya | nak sd aja paham ini uu sampah pak jokowi otak udah gak bisa mikir ya atau emang gak ada otak presiden gak ada beban ya gini nih mositidakpercaya |

### 4.6.3 Tokenisasi

Tahap ketiga adalah tokenisasi, yaitu mengubah semua kata menjadi bentuk token. Tahap tokenisasi telah disajikan dalam Tabel 4.7 dan Tabel 4.8.

**Tabel 4.7 Hasil Tokenisasi pada Data Latih**

| No | Hasil *Case Folding* | Hasil Tokenisasi |
|---|---|---|
| 1 | qaillaasyiqah semuga allah segera merobek robek kekuasaanya jokowi dan jajarannya | ['qaillaasyiqah', 'semuga', 'allah', 'segera', 'merobek', 'robek', 'kekuasaanya', 'jokowi', 'dan', 'jajarannya'] |
| 2 | pasti bangga dong dgn kinerja presiden jokowi yg sdh berhasil dan bikin spt ini indonesiamaju | ['pasti', 'bangga', 'dong', 'dgn', 'kinerja', 'presiden', 'jokowi', 'yg', 'sdh', 'berhasil', 'dan', 'bikin', 'spt', 'ini', 'indonesiamaju'] |
| 3 | aset negara yg najis haram jadah pantes negerimu diambang kehancuran kyai marufamin jokowi dpr ri | ['aset', 'negara', 'yg', 'najis', 'haram', 'jadah', 'pantes', 'negerimu', 'diambang', 'kehancuran', 'kyai', 'marufamin', 'jokowi', 'dpr', 'ri'] |
| 4 | terima pak jokowi dan basuki btp atas sumbangsihnya dalam pembangunan kota jakarta hingga di nobatkan sebagai kota terbaik dunia | ['terima', 'pak', 'jokowi', 'dan', 'basuki', 'btp', 'atas', 'sumbangsihnya', 'dalam', 'pembangunan', 'kota', 'jakarta', 'hingga', 'di', 'nobatkan', 'sebagai', 'kota', 'terbaik', 'dunia'] |
| 5 | selamat hari pahlawan terimakasih pak jokowi | ['selamat', 'hari', 'pahlawan', 'terimakasih', 'pak', 'jokowi'] |
| 6 | jokowi basuki btp aniesbaswedan pak boss gimana rasanya sakit hati kala melihat pak anies baswedan penuh prestasi sedangkan junjunan anda apa prestasinya nol besar apalagi si ahok yg dulu sering ngomong tai | ['jokowi', 'basuki', 'btp', 'aniesbaswedan', 'pak', 'boss', 'gimana', 'rasanya', 'sakit', 'hati', 'kala', 'melihat', 'pak', 'anies', 'baswedan', 'penuh', 'prestasi', 'sedangkan', 'junjunan', 'anda', 'apa', 'prestasinya', 'nol', 'besar', 'apalagi', 'si', 'ahok', 'yg', 'dulu', 'sering', 'ngomong', 'tai'] |

**Tabel 4.8 Hasil Tokenisasi pada Data Uji**

| No | Hasil *Case Folding* | Hasil Tokenisasi |
|---|---|---|
| 1 | presiden jokowi mengatakan dalam keberagaman perlu terus dilakukan untuk saling toleransi dan solidaritas guna pondasi yg kokoh dalam hidup dalam keberagaman indonesiamaju jokowi | ['presiden', 'jokowi', 'mengatakan', 'dalam', 'keberagaman', 'perlu', 'terus', 'dilakukan', 'untuk', 'saling', 'toleransi', 'dan', 'solidaritas', 'guna', 'pondasi', 'yg', 'kokoh', 'dalam', 'hidup', 'dalam', 'keberagaman', 'indonesiamaju', 'jokowi'] |
| 2 | nak sd aja paham ini uu sampah pak jokowi otak udah gak bisa mikir ya atau emang gak ada otak presiden gak ada beban ya gini nih mositidakpercaya | ['nak', 'sd', 'aja', 'paham', 'ini', 'uu', 'sampah', 'pak', 'jokowi', 'otak', 'udah', 'gak', 'bisa', 'mikir', 'ya', 'atau', 'emang', 'gak', 'ada', 'otak', 'presiden', 'gak', 'ada', 'beban', 'ya', 'gini', 'nih', 'mositidakpercaya'] |

### *4.6.4 Filtering*

Tahap keempat adalah *filtering,* yaitu menghapus *stopwords* yang ada dalam dokumen. Tahap *filtering* telah disajikan dalam Tabel 4.9 dan Tabel 4.10.

**Tabel 4.9 Hail Filtering pada Data Latih**

| No | Hasil Tokenisasi | Hasil *Filtering* |
|---|---|---|
| 1 | ['qaillaasyiqah', 'semuga', 'allah', 'segera', 'merobek', 'robek', 'kekuasaanya', 'jokowi', 'dan', 'jajarannya'] | ['qaillaasyiqah', 'semuga', 'allah', 'merobek', 'robek', 'kekuasaanya', 'jokowi', 'jajarannya'] |
| 2 | ['pasti', 'bangga', 'dong', 'dgn', 'kinerja', 'presiden', 'jokowi', 'yg', 'sdh', 'berhasil', 'dan', 'bikin', 'spt', 'ini', 'indonesiamaju'] | ['bangga', 'dgn', 'kinerja', 'presiden', 'jokowi', 'yg', 'sdh', 'berhasil', 'bikin', 'spt', 'indonesiamaju'] |
| 3 | ['aset', 'negara', 'yg', 'najis', 'haram', 'jadah', 'pantes', 'negerimu', 'diambang', 'kehancuran', 'kyai', 'marufamin', 'jokowi', 'dpr', 'ri'] | ['aset', 'negara', 'yg', 'najis', 'haram', 'jadah', 'pantes', 'negerimu', 'diambang', 'kehancuran', 'kyai', 'marufamin', 'jokowi', 'dpr', 'ri'] |
| 4 | ['terima', 'pak', 'jokowi', 'dan', 'basuki', 'btp', 'atas', 'sumbangsihnya', 'dalam', 'pembangunan', 'kota', 'jakarta', | ['terima', 'jokowi', 'basuki', 'btp', 'sumbangsihnya', 'pembangunan', 'kota', 'jakarta', 'nobatkan', 'kota', 'terbaik', 'dunia'] |

| | | |
|---|---|---|
| | 'hingga', 'di', 'nobatkan', 'sebagai', 'kota', 'terbaik', 'dunia'] | |
| 5 | ['selamat', 'hari', 'pahlawan', 'terimakasih', 'pak', 'jokowi'] | ['selamat', 'pahlawan', 'terimakasih', 'jokowi'] |
| 6 | ['jokowi', 'basuki', 'btp', 'aniesbaswedan', 'pak', 'boss', 'gimana', 'rasanya', 'sakit', 'hati', 'kala', 'melihat', 'pak', 'anies', 'baswedan', 'penuh', 'prestasi', 'sedangkan', 'junjunan', 'anda', 'apa', 'prestasinya', 'nol', 'besar', 'apalagi', 'si', 'ahok', 'yg', 'dulu', 'sering', 'ngomong', 'tai'] | ['jokowi', 'basuki', 'btp', 'aniesbaswedan', 'boss', 'gimana', 'sakit', 'hati', 'anies', 'baswedan', 'penuh', 'prestasi', 'junjunan', 'prestasinya', 'nol', 'si', 'ahok', 'yg', 'ngomong', 'tai'] |

**Tabel 4.10 Hasil Filtering pada Data Uji**

| No | Hasil Tokenisasi | Hasil *Filtering* |
|---|---|---|
| 1 | ['presiden', 'jokowi', 'mengatakan', 'dalam', 'keberagaman', 'perlu', 'terus', 'dilakukan', 'untuk', 'saling', 'toleransi', 'dan', 'solidaritas', 'guna', 'pondasi', 'yg', 'kokoh', 'dalam', 'hidup', 'dalam', 'keberagaman', 'indonesiamaju', 'jokowi'] | ['presiden', 'jokowi', 'keberagaman', 'toleransi', 'solidaritas', 'pondasi', 'yg', 'kokoh', 'hidup', 'keberagaman', 'indonesiamaju', 'jokowi'] |
| 2 | ['nak', 'sd', 'aja', 'paham', 'ini', 'uu', 'sampah', 'pak', 'jokowi', 'otak', 'udah', 'gak', 'bisa', 'mikir', 'ya', 'atau', 'emang', 'gak', 'ada', 'otak', 'presiden', 'gak', 'ada', 'beban', 'ya', 'gini', 'nih', 'mositidakpercaya'] | ['nak', 'sd', 'aja', 'paham', 'uu', 'sampah', 'jokowi', 'otak', 'udah', 'gak', 'mikir', 'ya', 'emang', 'gak', 'otak', 'presiden', 'gak', 'beban', 'ya', 'gini', 'nih', 'mositidakpercaya'] |

### *4.6.5 Stemming*

Tahap kelima adalah *stemming,* yaitu menghapus imbuhan atau mengubah semua kata menjadi kata dasar kata tersebut. Tahap *stemming* telah disajikan dalam Tabel 4.11 dan tabel Tabel 4.12.

**Tabel 4.11 Hasil *Stemming* pada Data Latih**

| No | Hasil Filtering | Hasil *Stemming* |
|---|---|---|
| 1 | ['qaillaasyiqah', 'semuga', 'allah', 'merobek', 'robek', 'kekuasaanya', 'jokowi', 'jajarannya'] | ['qaillaasyiqah', 'semuga', 'allah', 'robek', 'robek', 'kekuasaanya', 'jokowi', 'jajar'] |

| | | |
|---|---|---|
| 2 | ['bangga', 'dgn', 'kinerja', 'presiden', 'jokowi', 'yg', 'sdh', 'berhasil', 'bikin', 'spt', 'indonesiamaju'] | ['bangga', 'dgn', 'kerja', 'presiden', 'jokowi', 'yg', 'sdh', 'hasil', 'bikin', 'spt', 'indonesiamaju'] |
| 3 | ['aset', 'negara', 'yg', 'najis', 'haram', 'jadah', 'pantes', 'negerimu', 'diambang', 'kehancuran', 'kyai', 'marufamin', 'jokowi', 'dpr', 'ri'] | ['aset', 'negara', 'yg', 'najis', 'haram', 'jadah', 'pantes', 'negeri', 'ambang', 'hancur', 'kyai', 'marufamin', 'jokowi', 'dpr', 'ri'] |
| 4 | ['terima', 'jokowi', 'basuki', 'btp', 'sumbangsihnya', 'pembangunan', 'kota', 'jakarta', 'nobatkan', 'kota', 'terbaik', 'dunia'] | ['terima', 'jokowi', 'basuki', 'btp', 'sumbangsih', 'bangun', 'kota', 'jakarta', 'nobat', 'kota', 'baik', 'dunia'] |
| 5 | ['selamat', 'pahlawan', 'terimakasih', 'jokowi'] | ['selamat', 'pahlawan', 'terimakasih', 'jokowi'] |
| 6 | ['jokowi', 'basuki', 'btp', 'aniesbaswedan', 'boss', 'gimana', 'sakit', 'hati', 'anies', 'baswedan', 'penuh', 'prestasi', 'junjunan', 'prestasinya', 'nol', 'si', 'ahok', 'yg', 'ngomong', 'tai'] | ['jokowi', 'basuki', 'btp', 'aniesbaswedan', 'boss', 'gimana', 'sakit', 'hati', 'anies', 'baswedan', 'penuh', 'prestasi', 'junjunan', 'prestasi', 'nol', 'si', 'ahok', 'yg', 'ngomong', 'tai'] |

**Tabel 4.12 Hasil Stemming pada Data Uji**

| No | Hasil Filtering | Hasil *Stemming* |
|---|---|---|
| 1 | ['presiden', 'jokowi', 'keberagaman', 'toleransi', 'solidaritas', 'pondasi', 'yg', 'kokoh', 'hidup', 'keberagaman', 'indonesiamaju', 'jokowi'] | ['presiden', 'jokowi', 'agam', 'toleransi', 'solidaritas', 'pondasi', 'yg', 'kokoh', 'hidup', 'agam', 'indonesiamaju', 'jokowi'] |
| 2 | ['nak', 'sd', 'aja', 'paham', 'uu', 'sampah', 'jokowi', 'otak', 'udah', 'gak', 'mikir', 'ya', 'emang', 'gak', 'otak', 'presiden', 'gak', 'beban', 'ya', 'gini', 'nih', 'mositidakpercaya'] | ['nak', 'sd', 'aja', 'paham', 'uu', 'sampah', 'jokowi', 'otak', 'udah', 'gak', 'mikir', 'ya', 'emang', 'gak', 'otak', 'presiden', 'gak', 'beban', 'ya', 'gin', 'nih', 'mositidakpercaya'] |

### 4.6.6 Menentukan Fitur Kata

Selanjutnya untuk mendapatkan bobot dari TF-IDF diperlukan penentuan fitur kata terlebih dahulu. Fitur kata yang digunakan adalah token dari hasil *stemming*. Tabel daftar fitur telah disajikan dalam.

Tabel 4.13.

**Tabel 4.13 Daftar Fitur Kata**

| No | Fitur | No | Fitur | No | Fitur |
|---|---|---|---|---|---|
| 1 | jokowi | 32 | basuki | 63 | ngomong |
| 2 | qaillaasyiqah | 33 | btp | 64 | tai |
| 3 | semuga | 34 | sumbangsih | 65 | agam |
| 4 | allah | 35 | bangun | 66 | toleransi |
| 5 | robek | 36 | kota | 67 | solidaritas |
| 6 | kekuasaanya | 37 | jakarta | 68 | pondasi |
| 7 | jajar | 38 | nobat | 69 | kokoh |
| 8 | bangga | 39 | baik | 70 | hidup |
| 9 | dgn | 40 | dunia | 71 | nak |
| 10 | kerja | 41 | selamat | 72 | sd |
| 11 | presiden | 42 | pahlawan | 73 | aja |
| 12 | yg | 43 | terimakasih | 74 | paham |
| 13 | sdh | 44 | aniesbaswedan | 75 | uu |
| 14 | hasil | 45 | boss | 76 | sampah |
| 15 | bikin | 46 | gimana | 77 | otak |
| 16 | spt | 47 | sakit | 78 | udah |
| 17 | indonesiamaju | 48 | hati | 79 | gak |
| 18 | aset | 49 | anies | 80 | mikir |
| 19 | negara | 50 | baswedan | 81 | ya |
| 20 | najis | 51 | penuh | 82 | emang |
| 21 | haram | 52 | prestasi | 83 | beban |
| 22 | jadah | 53 | junjunan | 84 | gin |
| 23 | pantes | 54 | nol | 85 | nih |
| 24 | negeri | 55 | si | 86 | mositidakpercaya |
| 25 | ambang | 56 | ahok | 87 | ngomong |
| 26 | hancur | 57 | ngomong | 88 | tai |
| 27 | kyai | 58 | tai | 89 | agam |
| 28 | marufamin | 59 | agam | 90 | toleransi |
| 29 | dpr | 60 | toleransi | | |
| 30 | ri | 61 | si | | |
| 31 | terima | 62 | ahok | | |

### 4.6.7 Hitung Nilai Term Frequency (TF)

Tahap selanjutnya adalah melakukan perhitungan Term Frequency (TF). Tabel nilai TF dapat dilihat dalam Tabel 4.14.

**Tabel 4.14 Tabel Term Frequency**

| Fitur | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
|---|---|---|---|---|---|---|---|---|
| jokowi | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |
| qaillaasyiqah | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| semuga | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| allah | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| robek | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| kekuasaanya | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| jajar | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bangga | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| dgn | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| kerja | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| presiden | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| yg | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| sdh | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| hasil | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| bikin | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| spt | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| indonesiamaju | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| aset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| negara | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| najis | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| haram | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| jadah | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| pantes | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| negeri | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ambang | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| hancur | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| kyai | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| marufamin | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| dpr | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ri | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| terima | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| basuki | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| btp | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| sumbangsih | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| bangun | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| kota | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| jakarta | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| nobat | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| baik | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| dunia | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| selamat | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| pahlawan | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| terimakasih | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| aniesbaswedan | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| boss | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| gimana | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| sakit | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| hati | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| anies | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| baswedan | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| penuh | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| prestasi | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| junjunan | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| nol | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| si | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ahok | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ngomong | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| tai | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| agam | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| toleransi | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| solidaritas | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| pondasi | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| kokoh | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| hidup | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| nak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| sd | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| aja | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| paham | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| uu | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| sampah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| otak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| udah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| gak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| mikir | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| ya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| emang | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| beban | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| gin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| nih | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| mositidakpercaya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### 4.6.8 Hitung Nilai Wtf

Tahap selanjutnya adalah menghitung *Weight* Term Frequency (TF). Tabel nilai Wtf dapat dilihat dalam Tabel 4.15.

**Tabel 4.15 Tabel Nilai Wtf**

| Fitur | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
|---|---|---|---|---|---|---|---|---|
| jokowi | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,30 | 1,00 |
| qaillaasyiqah | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| semuga | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| allah | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| robek | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| kekuasaanya | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| jajar | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bangga | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| dgn | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| kerja | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| presiden | 0 | 1,00 | 0 | 0 | 0 | 0 | 1,00 | 1,00 |
| yg | 0 | 1,00 | 1,00 | 0 | 0 | 1,00 | 1,00 | 0 |
| sdh | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| hasil | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| bikin | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| spt | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 | 0 |
| indonesiamaju | 0 | 1,00 | 0 | 0 | 0 | 0 | 1,00 | 0 |
| aset | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| negara | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| najis | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| haram | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| jadah | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| pantes | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| negeri | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| ambang | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| hancur | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| kyai | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| marufamin | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| dpr | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| ri | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 | 0 |
| terima | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |
| basuki | 0 | 0 | 0 | 1,00 | 0 | 1,00 | 0 | 0 |
| btp | 0 | 0 | 0 | 1,00 | 0 | 1,00 | 0 | 0 |
| sumbangsih | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |
| bangun | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |
| kota | 0 | 0 | 0 | 1,30 | 0 | 0 | 0 | 0 |
| jakarta | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| nobat | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |
| baik | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |
| dunia | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 | 0 |
| selamat | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 |
| pahlawan | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 |
| terimakasih | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 | 0 |
| aniesbaswedan | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| boss | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| gimana | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| sakit | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| hati | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| anies | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| baswedan | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| penuh | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| prestasi | 0 | 0 | 0 | 0 | 0 | 1,30 | 0 | 0 |
| junjunan | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| nol | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| si | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| ahok | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| ngomong | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| tai | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 | 0 |
| agam | 0 | 0 | 0 | 0 | 0 | 0 | 1,30 | 0 |
| toleransi | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 |
| solidaritas | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 |
| pondasi | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 |
| kokoh | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 |
| hidup | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 | 0 |
| nak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| sd | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| aja | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| paham | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| uu | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| sampah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| otak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,30 |
| udah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| gak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,48 |
| mikir | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| ya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,30 |
| emang | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| beban | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| gin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| nih | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |
| mositidakpercaya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1,00 |

### 4.6.9 Hitung Nilai DF dan IDF

Proses selanjutnya adalah menghitung Document Frequency (DF) dan *inverted* Document Frequency (IDF). Tabel nilai IDF telah disajikan dalam Tabel 4.16.

**Tabel 4.16 Tabel Nilai IDF**

| Fitur | IDF |
|---|---|
| jokowi | 0 |
| qaillaasyiqah | 0,90 |
| semuga | 0,90 |
| allah | 0,90 |
| robek | 0,90 |
| kekuasaanya | 0,90 |
| jajar | 0,90 |
| bangga | 0,90 |
| dgn | 0,90 |
| kerja | 0,90 |
| presiden | 0,43 |
| yg | 0,30 |
| sdh | 0,90 |
| hasil | 0,90 |
| bikin | 0,90 |
| spt | 0,90 |
| indonesiamaju | 0,60 |
| aset | 0,90 |
| negara | 0,90 |
| najis | 0,90 |
| haram | 0,90 |
| jadah | 0,90 |
| pantes | 0,90 |
| negeri | 0,90 |
| ambang | 0,90 |
| hancur | 0,90 |
| kyai | 0,90 |
| marufamin | 0,90 |
| dpr | 0,90 |
| ri | 0,90 |
| terima | 0,90 |
| basuki | 0,60 |
| btp | 0,60 |
| sumbangsih | 0,90 |
| bangun | 0,90 |
| kota | 0,90 |

| | |
|---|---|
| jakarta | 0,90 |
| nobat | 0,90 |
| baik | 0,90 |
| dunia | 0,90 |
| selamat | 0,90 |
| pahlawan | 0,90 |
| terimakasih | 0,90 |
| aniesbaswedan | 0,90 |
| boss | 0,90 |
| gimana | 0,90 |
| sakit | 0,90 |
| hati | 0,90 |
| anies | 0,90 |
| baswedan | 0,90 |
| penuh | 0,90 |
| prestasi | 0,90 |
| junjunan | 0,90 |
| nol | 0,90 |
| si | 0,90 |
| ahok | 0,90 |
| ngomong | 0,90 |
| tai | 0,90 |
| agam | 0 |
| toleransi | 0 |
| solidaritas | 0 |
| pondasi | 0 |
| kokoh | 0 |
| hidup | 0 |
| nak | 0 |
| sd | 0 |
| aja | 0 |
| paham | 0 |
| uu | 0 |
| sampah | 0 |
| otak | 0 |
| udah | 0 |
| gak | 0 |
| mikir | 0 |
| ya | 0 |
| emang | 0 |
| beban | 0 |
| gin | 0 |
| nih | 0 |

| | |
|---|---|
| mositidakpercaya | 0 |

### 4.6.10 Hitung Nilai TF-IDF

Terakhir adalah menghitung TF-IDF. Tabel nilai TF-IDF telah disajikan dalam Tabel 4.17.

**Tabel 4.17 Tabel Nilai TF-IDF**

| Fitur | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
|---|---|---|---|---|---|---|---|---|
| jokowi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| qaillaasyiqah | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| semuga | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| allah | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| robek | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| kekuasaanya | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| jajar | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bangga | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| dgn | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| kerja | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| presiden | 0 | 0,43 | 0 | 0 | 0 | 0 | 0,43 | 0,43 |
| yg | 0 | 0,30 | 0,30 | 0 | 0 | 0,30 | 0,30 | 0 |
| sdh | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| hasil | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| bikin | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| spt | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| indonesiamaju | 0 | 0,60 | 0 | 0 | 0 | 0 | 0,60 | 0 |
| aset | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| negara | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| najis | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| haram | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| jadah | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| pantes | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| negeri | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| ambang | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| hancur | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| kyai | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| marufamin | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| dpr | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| ri | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| terima | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| basuki | 0 | 0 | 0 | 0,60 | 0 | 0,60 | 0 | 0 |
| btp | 0 | 0 | 0 | 0,60 | 0 | 0,60 | 0 | 0 |
| sumbangsih | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| bangun | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| kota | 0 | 0 | 0 | 1,17 | 0 | 0 | 0 | 0 |
| jakarta | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| nobat | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| baik | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| dunia | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| selamat | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 |
| pahlawan | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 |
| terimakasih | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 |
| aniesbaswedan | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| boss | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| gimana | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| sakit | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| hati | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| anies | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| baswedan | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| penuh | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| prestasi | 0 | 0 | 0 | 0 | 0 | 1,17 | 0 | 0 |
| junjunan | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| nol | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| si | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| ahok | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| ngomong | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| tai | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| agam | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| toleransi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| solidaritas | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pondasi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| kokoh | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hidup | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| nak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| sd | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| aja | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| paham | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| uu | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| sampah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| otak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| udah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| mikir | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| emang | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| beban | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| nih | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| mositidakpercaya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *4.6.11 Lexicon Based Feature*

Selanjutnya dilakukan perhitungan bobot *Lexicon Based Feature.* Tabel nilai *Lexicon Based Feature* telah disajikan dalam Tabel 4.18.

**Tabel 4.18 Tabel *Lexicon Based Feature***

| Kode Fitur | Dokumen | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
| F1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F2 | 0 | 2 | 0 | 2 | 3 | 1 | 2 | 0 |
| F3 | 1 | 0 | 4 | 0 | 0 | 2 | 0 | 2 |
| F4 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| F5 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 |
| F6 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| F7 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| F8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F10 | 0 | 0,5 | 0 | 0,5 | 0,33 | 0 | 0 | 0 |
| F11 | 1 | 0 | 0,75 | 0 | 0 | 0,5 | 0 | 0 |
| F12 | 0 | 0,5 | 0 | 0,5 | 0 | 0 | 0 | 0 |
| F13 | 0 | 0 | 0,5 | 0 | 0 | 0 | 0 | 0 |
| F14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Kelas | -1 | 1 | -1 | 1 | 1 | -1 | 1 | -1 |

### *4.6.12* Support Vector Machine

Proses selanjutnya adalah penerapan Support Vector Machine. Penerapan SVM dimulai dengan penggabungan tabel pada fitur dari bobot TF-IDF dengan *Lexicon Based Feature*. Fitur gabungan telah disajikan dalam Tabel 4.19.

**Tabel 4.19 Tabel Fitur Gabungan**

| Fitur | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
|---|---|---|---|---|---|---|---|---|
| jokowi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| qaillaasyiqah | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| semuga | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| allah | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| robek | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| kekuasaanya | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| jajar | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| bangga | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| dgn | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| kerja | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| presiden | 0 | 0,43 | 0 | 0 | 0 | 0 | 0,43 | 0,43 |
| yg | 0 | 0,30 | 0,30 | 0 | 0 | 0,30 | 0,30 | 0 |
| sdh | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| hasil | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| bikin | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| spt | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 | 0 |
| indonesiamaju | 0 | 0,60 | 0 | 0 | 0 | 0 | 0,60 | 0 |
| aset | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| negara | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| najis | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| haram | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| jadah | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| pantes | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| negeri | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| ambang | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| hancur | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| kyai | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| marufamin | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| dpr | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| ri | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 | 0 |
| terima | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| basuki | 0 | 0 | 0 | 0,60 | 0 | 0,60 | 0 | 0 |
| btp | 0 | 0 | 0 | 0,60 | 0 | 0,60 | 0 | 0 |
| sumbangsih | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| bangun | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| kota | 0 | 0 | 0 | 1,17 | 0 | 0 | 0 | 0 |
| jakarta | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| nobat | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| baik | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| dunia | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 | 0 |
| selamat | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 |
| pahlawan | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 |
| terimakasih | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 0 |
| aniesbaswedan | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| boss | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| gimana | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| sakit | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| hati | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| anies | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| baswedan | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| penuh | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| prestasi | 0 | 0 | 0 | 0 | 0 | 1,17 | 0 | 0 |
| junjunan | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| nol | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| si | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| ahok | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| ngomong | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| tai | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 |
| agam | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| toleransi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| solidaritas | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pondasi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| kokoh | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hidup | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| nak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| sd | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| aja | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| paham | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| uu | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| sampah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| otak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| udah | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gak | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| mikir | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| emang | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| beban | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gin | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| nih | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| mositidakpercaya | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F2 | 0 | 2 | 0 | 2 | 3 | 1 | 2 | 0 |
| F3 | 1 | 0 | 4 | 0 | 0 | 2 | 0 | 2 |
| F4 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| F5 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 |
| F6 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| F7 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| F8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F10 | 0 | 0,5 | 0 | 0,5 | 0,33 | 0 | 0 | 0 |
| F11 | 1 | 0 | 0,75 | 0 | 0 | 0,5 | 0 | 0 |
| F12 | 0 | 0,5 | 0 | 0,5 | 0 | 0 | 0 | 0 |
| F13 | 0 | 0 | 0,5 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| F14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Setelah mendapatkan tabel gabungan fitur, langkah selanjutnya adalah menginisialisasi variabel atau parameter pada SVM. Tabel inisialisasi parameter SVM dapat dilihat dalam Tabel 4.20.

**Tabel 4.20 Parameter SVM**

| | |
|---|---|
| λ (lambda) | 0,5 |
| γ (gamma) | 0,001 |
| ϵ (epsilon) | 0,0001 |
| C (konstanta) | 1 |
| σ (sigma) | 5 |
| A (alpha) | 0 |
| itermax | 5 |

Setelah mendapatkan nilai parameter awal, langkah selanjutnya adalah perhitungan Kernel RBF. Tabel perhitungan Kernel RBF telah disajikan dalam Tabel 4.21.

**Tabel 4.21 Tabel Perhitungan Kernel RBF**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 0,661 | 0,518 | 0,641 | 0,664 | 0,660 |
| 2 | 0,661 | 1 | 0,345 | 0,753 | 0,801 | 0,565 |
| 3 | 0,518 | 0,345 | 1 | 0,334 | 0,346 | 0,473 |
| 4 | 0,641 | 0,753 | 0,334 | 1 | 0,778 | 0,563 |
| 5 | 0,664 | 0,801 | 0,346 | 0,778 | 1 | 0,590 |
| 6 | 0,660 | 0,565 | 0,473 | 0,563 | 0,590 | 1 |
| 7 | 0,778 | 0,849 | 0,407 | 0,803 | 0,901 | 0,666 |
| 8 | 0,851 | 0,717 | 0,563 | 0,691 | 0,715 | 0,725 |

Langkah selanjutnya adalah perhitungan matriks Hessian. Tabel perhitungan matrik Hessian telah disajikan dalam Tabel 4.22.

**Tabel 4.22 Tabel Perhitungan matrik Hessian**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1,25 | -0,91 | 0,77 | -0,89 | -0,91 | 0,91 |
| 2 | -0,911 | 1,25 | -0,595 | 1,003 | 1,051 | -0,815 |
| 3 | 0,768 | -0,595 | 1,25 | -0,584 | -0,596 | 0,723 |
| 4 | -0,891 | 1,003 | -0,584 | 1,25 | 1,028 | -0,813 |
| 5 | -0,914 | 1,051 | -0,596 | 1,028 | 1,25 | -0,840 |
| 6 | 0,910 | -0,815 | 0,723 | -0,813 | -0,840 | 1,25 |

Selanjutnya dilakukan *Sequential Training* dengan melakukan perulangan sebanyak iter max. Pada iterasi pertama, tabel perhitungan nilai E telah diuraikan dalam Tabel 4.23, tabel perhitungan nilai delta alpha telah diuraikan dalam Tabel 4.24, dan tabel perhitungan nilai alpha telah diuraikan dalam Tabel 4.25.

**Tabel 4.23 Tabel Perhitungan nilai Error pada iterasi ke-1**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ei | 0 | 0 | 0 | 0 | 0 | 0 |

**Tabel 4.24 Tabel Perhitungan nilai Delta Alpha pada iterasi ke-1**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| δα | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |

**Tabel 4.25 Tabel Perhitungan nilai Alpha pada iterasi ke-1**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| α lama | 0 | 0 | 0 | 0 | 0 | 0 |
| α baru | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |

Pada iterasi kedua, tabel perhitungan nilai E dapat dilihat pada Tabel 4.26, tabel perhitungan nilai delta alpha dapat dilihat pada Tabel 4.27, dan tabel perhitungan nilai alpha dapat dilihat pada Tabel 4.28.

**Tabel 4.26 Tabel Perhitungan nilai Error pada iterasi ke-2**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ei | 0,000212 | 0,000984 | 0,000965 | 0,0009932 | 0,00098 | 0,000415492 |

**Tabel 4.27 Tabel Perhitungan nilai Delta Alpha pada iterasi ke-2**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| δα | 0,001 | 0,000999 | 0,000999 | 0,000999 | 0,001 | 0,000999585 |

**Tabel 4.28 Tabel Perhitungan nilai Alpha pada iterasi ke-2**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| α lama | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| α baru | 0,002 | 0,001999 | 0,001999 | 0,001999 | 0,001999 | 0,001999585 |

Pada iterasi ketiga, tabel perhitungan nilai E dapat dilihat pada Tabel 4.29, tabel perhitungan nilai delta alpha dapat dilihat pada Tabel 4.30, dan tabel perhitungan nilai alpha dapat dilihat pada Tabel 4.31.

**Tabel 4.29 Tabel Perhitungan nilai Error pada iterasi ke-3**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ei | 0,000425 | 0,001965 | 0,001931 | 0,0019843 | 0,0019574 | 0,000832007 |

**Tabel 4.30 Tabel Perhitungan nilai Delta Alpha pada iterasi ke-3**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| δα | 0,001 | 0,000998 | 0,000998 | 0,000998 | 0,000998 | 0,000999168 |

**Tabel 4.31 Tabel Perhitungan nilai Alpha pada iterasi ke-3**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| α lama | 0,002 | 0,001999 | 0,001999 | 0,001999 | 0,001999 | 0,001999585 |
| α baru | 0,002999 | 0,002997 | 0,002997 | 0,002997 | 0,0029971 | 0,002998753 |

Pada iterasi keempat, tabel perhitungan nilai E dapat dilihat pada Tabel 4.32, tabel perhitungan nilai delta alpha dapat dilihat pada Tabel 4.33, dan tabel perhitungan nilai alpha dapat dilihat pada Tabel 4.34.

**Tabel 4.32 Tabel Perhitungan nilai Error pada iterasi ke-4**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ei | 0,00064 | 0,002945 | 0,002897 | 0,0029732 | 0,0029329 | 0,001249535 |

**Tabel 4.33 Tabel Perhitungan nilai Delta Alpha pada iterasi ke-4**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| δα | 0,000999 | 0,000997 | 0,000997 | 0,000997 | 0,0009971 | 0,00099875 |

**Tabel 4.34 Tabel Perhitungan nilai Alpha pada iterasi ke-4**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| α lama | 0,002999 | 0,002997 | 0,002997 | 0,002997 | 0,0029971 | 0,002998753 |
| α baru | 0,003999 | 0,003994 | 0,003994 | 0,003994 | 0,0039941 | 0,003997503 |

Pada iterasi kelima, tabel perhitungan nilai E dapat dilihat pada Tabel 4.35, tabel perhitungan nilai delta alpha dapat dilihat pada Tabel 4.36, dan tabel perhitungan nilai alpha dapat dilihat pada Tabel 4.37.

**Tabel 4.35 Tabel Perhitungan nilai Error pada iterasi ke-5**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ei | 0,000856 | 0,003922 | 0,003862 | 0,00396 | 0,0039062 | 0,001668069 |

**Tabel 4.36 Tabel Perhitungan nilai Delta Alpha pada iterasi ke-5**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| δα | 0,000999 | 0,000996 | 0,000996 | 0,000996 | 0,0009961 | 0,000998332 |

**Tabel 4.37 Tabel Perhitungan nilai Alpha pada iterasi ke-5**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| α lama | 0,003999 | 0,003994 | 0,00399420 | 0,003994 | 0,0039941 | 0,00399750 |
| α baru | 0,0049978 | 0,0049901 | 0,0049903 | 0,0049900 | 0,0049902 | 0,004995 |

Langkah selanjutnya adalah menginisialisasi variabel Support Vector (SV) dengan nilai sama dengan nilai α pada iterasi terakhir. Tabel nilai Support Vector (SV) dapat dilihat pada Tabel 4.38.

**Tabel 4.38 Tabel Nilai Support Vector (SV)**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| SV | 0,0049978 | 0,0049901 | 0,0049903 | 0,0049900 | 0,0049902 | 0,004995 |
| Kelas | -1 | 1 | -1 | 1 | 1 | -1 |

Kemudian untuk dapat menghitung nilai bias, diperlukan pencarian nilai SV terbesar dari kelas positif dan kelas negatif. Pada tabel dapat dilihat dokumen ke-1 memiliki nilai Support Vector terbesar di antara kelas negatif lain. Dokumen ke-5 memiliki nilai Support Vector terbesar di antara kelas positif lain. Oleh sebab itu nilai dari $K(x_i, x^-)$ di wakili oleh dokumen ke-1 dan nilai dari $K(x_i, x^+)$ diwakili oleh dokumen ke-5. Tabel nilai $K(x_i, x^-)$ dan $K(x_i, x^+)$ dapat dilihat pada Tabel 4.39.

**Tabel 4.39 Tabel Nilai K(x,x$^-$) dan K(x,x$^+$)**

| $x_i$ | $K(x_i, x^-)$ | $K(x_i, x^+)$ |
|---|---|---|
| 1 | 1 | 0,664 |
| 2 | 0,661 | 0,801 |
| 3 | 0,518 | 0,346 |
| 4 | 0,641 | 0,778 |

| | | |
|---|---|---|
| 5 | 0,664 | 1 |
| 6 | 0,660 | 0,590 |

Setelah mendapat nilai $K(x_i, x^-)$ dan $K(x_i, x^+)$, langkah selanjutnya adalah melakukan perhitungan nilai bias. Perhitungan nilai bias telah disajikan dalam Tabel 4.40.

**Tabel 4.40 Tabel Perhitungan Nilai Bias**

| $x_i$ | $w * x^-$ | $w * x^+$ |
|---|---|---|
| 1 | -0,005 | -0,00332 |
| 2 | 0,003296 | 0,003999 |
| 3 | -0,00258 | -0,00173 |
| 4 | 0,0032 | 0,003883 |
| 5 | 0,003315 | 0,00499 |
| 6 | -0,0033 | -0,00295 |
| Jumlah | -0,00107 | 0,004881 |
| bias | -0,00191 | |

Langkah terakhir adalah melakukan *testing* pada data uji dengan mencari nilai dari $f(x)$. Jika hasil dari nilai $f(x)$ lebih dari 0 maka dokumen tersebut termasuk ke dalam kelas 1 atau bukan perundungan, sebaliknya apabila hasil dari $f(x)$ kurang dari 0 maka dokumen tersebut masuk ke dalam kelas -1 atau perundungan. Hasil perhitungan *testing* telah diuraikan dalam Tabel 4.41.

**Tabel 4.41 Tabel Perhitungan Testing**

| $\alpha_i y_i K(x_i, x)$ | d7 | d8 |
|---|---|---|
| 1 | -0,00389 | -0,00425 |
| 2 | 0,004235 | 0,003576 |
| 3 | -0,00203 | -0,00281 |
| 4 | 0,004009 | 0,003447 |
| 5 | 0,004498 | 0,00357 |
| 6 | -0,00333 | -0,00362 |
| jumlah | 0,003493 | -0,0001 |
| $f(x)$ | 0,000904 | -0,002 |

## 4.7 Perancangan Pengujian

Pada saat melakukan pengujian validasi pada sistem, tingkat *accuracy*, *precision, recall, F-Measure* menentukan baik tidaknya kinerja algoritme. Proses pengujian parameter sebagai berikut:

1. Pengujian parameter σ (Sigma),
2. Pengujian parameter λ (Lambda),
3. Pengujian parameter γ (Gamma),
4. Pengujian parameter C (Complexity),
5. Pengujian parameter ε (Epsilon),
6. Pengujian parameter iter max.
7. Pengujian jenis kernel.
8. Pengujian pengaruh *Lexicon Based Feature*

Perancangan pengujian telah disajikan dalam Tabel 4.42 sampai Tabel 4.48.

**Tabel 4.42 Tabel Pengujian Parameter σ**

| Nilai σ (sigma) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 2 | | | | |
| 2,5 | | | | |
| 3 | | | | |
| 3,5 | | | | |
| 4 | | | | |
| 4,5 | | | | |
| 5 | | | | |
| 5,5 | | | | |
| 6 | | | | |

**Tabel 4.43 Tabel Pengujian Parameter λ**

| Nilai λ (lambda) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,1 | | | | |

| | | | | |
|---|---|---|---|---|
| 0,3 | | | | |
| 0,5 | | | | |
| 0,7 | | | | |
| 0,9 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

**Tabel 4.44 Tabel Pengujian Parameter γ**

| Nilai γ (gamma) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,000001 | | | | |
| 0,00001 | | | | |
| 0,0001 | | | | |
| 0,001 | | | | |
| 0,01 | | | | |
| 0,1 | | | | |
| 1 | | | | |

**Tabel 4.45 Tabel Pengujian Parameter c**

| Nilai c (Complexity) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,1 | | | | |
| 0,5 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 10 | | | | |

**Tabel 4.46 Tabel Pengujian Parameter ε**

| Nilai ε (epsilon) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,00001 | | | | |
| 0,0001 | | | | |
| 0,001 | | | | |
| 0,01 | | | | |
| 0,1 | | | | |
| 1 | | | | |

**Tabel 4.47 Tabel Pengujian Parameter iter Max**

| Nilai iter Max | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 10 | | | | |
| 20 | | | | |
| 30 | | | | |
| 40 | | | | |
| 50 | | | | |
| 60 | | | | |
| 70 | | | | |
| 80 | | | | |
| 90 | | | | |
| 100 | | | | |

**Tabel 4.48 Tabel Pengujian Jenis kernel**

| Jenis Kernel | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| Linear | | | | |
| Polynomial Degree d | | | | |
| Gaussian RBF | | | | |

**Tabel 4.49 Tabel Pengaruh *Lexicon Based Feature***

| Fitur | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| *Lexicon Based Feature* | | | | |
| TF-IDF | | | | |
| *Lexicon Based Feature* + TF-IDF | | | | |

# BAB 5 IMPLEMENTASI

Bab implementasi menjelaskan tentang implementasi pada sistem deteksi perundungan siber pada *tweet* berbahasa Indonesia menggunakan Support Vector Machine dengan *Lexicon Based Feature*. Tahapan-tahapan tersebut didasari oleh perancangan yang telah dijabarkan pada bab sebelumnya.

## 5.1 Implementasi Sistem

Implementasi sistem pada penelitian deteksi perundungan siber ini dilakukan melalui beberapa langkah, yaitu *preprocessing*, pembobotan TF-IDF, *Lexicon Based Feature*, implementasi Support Vector Machine, dan evaluasi. Implementasi method *main* dapat dilihat pada Kode Program 5.1.

```
1  def main():
2      hasiltrain = preprocessing(train)
3      hasiltest = preprocessing(test)
4      bobottfidf = tfidf(hasiltrain, hasiltest)
5      bobotlex = lexicon(doc_tweet, boosterwords, kamuspos, kamusneg,
   kbbi)
6      bobot = bobottfidf + bobotlex
7      liner = kernelLinear(bobot)
8      acc, preci, rec, fm = svm(liner)
9      return acc, preci, rec, fm
```

**Kode Program 5.1 Implementasi method *main***

Penjelasan Kode Program 5.1:

1. Baris 2 menjalankan method *preprocessing* dengan parameter *train* kemudian menyimpannya ke dalam variabel “hasiltrain”.
2. Baris 3 menjalankan method *preprocessing* dengan parameter *test* kemudian menyimpannya ke dalam variabel “hasiltest”.
3. Baris 4 menjalankan method tfidf dengan parameter hasiltrain dan hasiltest kemudian menyimpannya ke dalam variabel “bobottfidf”.
4. Baris 5 menjalankan method lexicon dengan parameter *doc_tweet*, *boosterwords,* kamuspos, kamusneg, dan KBBI kemudian menyimpannya ke dalam variabel “bobotlex”.
5. Baris 6 menggabungkan variabel bobottfidf dengan bobotlex kemudian menyimpannya ke dalam variabel “bobot”.
6. Baris 7 menjalankan method kernelLinear dengan variabel bobot kemudian menyimpannya ke dalam variabel “liner”.
7. Baris 8 menjalankan method svm dengan variabel liner kemudian menyimpannya ke dalam variabel “acc”, “preci”, “rec”, dan “fm”.
8. Baris 9 mengembalikan nilai dari variabel acc, preci, rec, dan fm.

## 5.2 *Preprocessing*

Tahap awal implementasi pada penelitian ini merupakan *preprocessing*, pada tahap ini dilakukan beberapa langkah yang bertujuan untuk mengubah dokumen *tweet* yang digunakan menjadi bentuk yang lebih mudah diolah. Langkah-langkah pada *preprocessing* adalah *cleaning, case folding*, tokenisasi, *filtering*, dan *stemming*.

### *5.2.1 Cleaning*

*Cleaning* adalah tahap untuk membersihkan dokumen *tweet* dengan cara menghilangkan angka, simbol, dan tautan. Implementasi pada tahapan *cleaning* telah diuraikan dalam Kode Program 5.2.

```
1  def cleaning(text):
2   cleaned = []
3   for i in range(len(text)):
4     text[i] = re.sub (r"http\S+", " ", text[i])
5     hasil1 = re.sub (r"([^a-zA-Z]+)", " " , text[i])
6     cleaned.append(hasil1)
7   return cleaned
```

**Kode Program 5.2 Implementasi *Cleaning***

Penjelasan Kode Program 5.2:

1. Baris 2 membuat list bernama "*cleaned*" yang berfungsi menampung hasil dari perhitungan regex.
2. Baris 3-6 melakukan perhitungan subtitusi regex dengan menghilangkan tautan terlebih dahulu kemudian mengambil semua kata yang ada, kemudian menyimpannya pada list *cleaned*.
3. Baris 7 mengembalikan nilai dari list *cleaned*.

### *5.2.2 Case Folding*

*Case Folding* adalah tahap untuk mengganti seluruh huruf pada data yang digunakan menjadi huruf kecil. Implementasi pada proses *case folding* telah diuraikan dalam Kode Program 5.3.

```
1  def casefolding(text):
2   foldedcase = []
3   for j in text:
4     temp = j.lower()
5     foldedcase.append(temp)
6   return foldedcase
```

**Kode Program 5.3 Implementasi *Case Folding***

Penjelasan Kode Program 5.3:

1. Baris 2 membuat list bernama "*foldedcase*" yang berfungsi menampung hasil dari operasi perubahan huruf menjadi huruf kecil.
2. Baris 3-5 mengganti seluruh huruf pada dokumen *tweet* menjadi huruf kecil kemudian menyimpan hasilnya pada list *foldedcase*.
3. Baris 6 mengembalikan nilai dari list *foldedcase*.

### 5.2.3 Tokenisasi

Tokenisasi adalah tahap untuk mengubah semua kalimat yang ada pada dokumen *tweet* menjadi *string* tunggal atau dapat disebut dengan token. Implementasi pada tahap tokenisasi dapat dilihat dalam Kode Program 5.4.

```
1  def token(text):
2    token = []
3    for j in text:
4      hasil1 = j.split()
5      token.append(hasil1)
6    return token
```

**Kode Program 5.4 Implementasi Tokenisasi**

Penjelasan Kode Program 5.4:

1. Baris 2 membuat list bernama "token" yang berfungsi menampung hasil dari perubahan huruf.
2. Baris 3-5 memisahkan semua kata pada dokumen *tweet* menjadi huruf kata-kata tunggal kemudian menyimpan hasilnya pada list token.
3. Baris 6 mengembalikan nilai dari list token.

### *5.2.4 Filtering*

*Filtering* adalah tahap untuk menghilangkan *stopwords* atau token-token yang tidak mengandung makna penting pada semua dokumen *tweet*. Implementasi pada proses *filtering* telah diuraikan pada Kode Program 5.5.

```
1  def filtering(text, stopwords):
2    filtered = []
3    for i in range(len(text)):
4      hasil1 = []
5      for token in text[i]:
6        if token not in stopwords:
7          hasil1.append(token)
8      filtered.append(hasil1)
9    return filtered
```

**Kode Program 5.5 Implementasi *Filtering***

Penjelasan Kode Program 5.5:

1. Baris 2 membuat list bernama "*filtered*" yang berfungsi menampung semua hasil seleksi token dari tiap dokumen.
2. Baris 4 menginisialisasi nilai kosong pada list bernama "temp" yang berfungsi menampung hasil seleksi token.
3. Baris 5-8 melakukan seleksi di tiap dokumen pada token dengan melakukan pengecekan apabila token tersebut tidak ada pada daftar stopwords. Kemudian memasukkan token pada temp, dan menyimpan list temp pada list filtered.
4. Baris 9 mengembalikan nilai dari list *filtered*.

### *5.2.5 Stemming*

*Stemming* merupakan tahap untuk mengganti token menjadi akar katanya. Implementasi pada proses *stemming* telah diuraikan pada Kode Program 5.6.

```
1  def stemming(text, stemmer):
2
3    for i in range(len(text)):
4      for token in range(len(text[i])):
5        text[i][token] = stemmer.stem(text[i][token])
6
7    return text
```

**Kode Program 5.6 Implementasi *Stemming***

Penjelasan Kode Program 5.6:

1. Baris 3-5 menggunakan library stemmer untuk mengubah tiap token yang ada pada dokumen menjadi kata dasar.
2. Baris 7 mengembalikan nilai dari text.

## 5.3 Pembobotan kata (TF-IDF)

Tahap kedua implementasi pada penelitian ini adalah pembobotan TF-IDF, tahap ini bertujuan untuk memberikan bobot pada tiap token. Langkah-langkah pada pembobotan TF-IDF yaitu, menentukan fitur kata, menghitung nilai Term Frequency (TF), menghitung nilai *Weight* Term Frequency (WTF), menghitung nilai Document Frequency DF dan Inverse Document Frequency IDF, dan menghitung nilai Term Frequency-Inverse Document Frequency TF-IDF.

### 5.3.1 Menentukan Fitur Kata

Tahap awal dari pembobotan kata adalah menentukan fitur terlebih dahulu. Fitur didapat dari tiap token di semua dokumen. Implementasi pada proses menentukan fitur kata telah disajikan pada Kode Program 5.7.

```
1  def fitur_list(text):
2    fitur =[]
3    for token in text:
4      for w in token:
5        if w not in fitur:
6          fitur.append(w)
7    return fitur
```

**Kode Program 5.7 Implementasi Menentukan Fitur Kata**

Penjelasan Kode Program 5.7:

1. Baris 2 membuat list bernama “fitur”.
2. Baris 3-6 menjabarkan setiap token pada tiap dokumen untuk dijadikan fitur dengan menambahkannya pada list fitur.
3. Baris 7 mengembalikan nilai dari fitur.

### *5.3.2* Term Frequency

Tahap selanjutnya dari pembobotan kata adalah menghitung nilai Term Frequency (TF). Nilai TF didapat dengan cara menghitung munculnya token di tiap dokumen. Implementasi pada proses menghitung nilai Term Frequency telah disajikan pada Kode Program 5.8.

```
1  def termfreq(fitur, text):
2      termfreq = []
3      for term in fitur:
4          temp = []
5          for doc in text:
6              temp.append(doc.count(term))
7          termfreq.append(temp)
8      return termfreq
```

**Kode Program 5.8 Implementasi Term Frequency**

Penjelasan Kode Program 5.8:

1. Baris 2 membuat list bernama “termfreq”.
2. Baris 4 menginisialisasi nilai dari list bernama “temp”
3. Baris 5-7 menghitung jumlah kemunculan sebuah term pada suatu dokumen kemudian menambahkan jumlah tersebut ke dalam temp, kemudian menambahkan nilai dari temp ke dalam termfreq.
4. Baris 8 mengembalikan nilai dari termfreq.

### *5.3.3 Weighted* Term Frequency

Tahap selanjutnya adalah menggunakan nilai TF untuk menghitung nilai dari *weighted* Term Frequency (WTF). Nilai WTF didapat dengan melakukan perhitungan dengan rumus yang telah diuraikan pada Persamaan 2.1. Implementasi pada proses menghitung nilai *weighted* Term Frequency dapat dilihat dalam Kode Program 5.9.

```
1  def weightTF(termfreq):
2      weight_tf = []
3      for i in range(len(termfreq)):
4          temp = []
5          for n in range(len(termfreq[i])):
6              wtf = 0
7              if termfreq[i][n]>0:
8                  wtf = 1+ math.log10(termfreq[i][n])
9              else:
10                 wtf = 0
11             temp.append(wtf)
12         weight_tf.append(temp)
13     return weight_tf
```

**Kode Program 5.9 Implementasi *Weighted* Term Frequency**

Penjelasan Kode Program 5.9:

1. Baris 2 membuat list bernama “weight_tf”.
2. Baris 4 menginisialisasi nilai dari list bernama “temp”

3. Baris 6-10 melakukan seleksi kondisi. Jika nilai Term Frequency dari token lebih dari nol, dilakukan perhitungan rumus wtf dan memasukkan nilainya ke dalam variabel wtf, dan jika nilai Term Frequency nol, nilai wtf bernilai nol.
4. Baris 11-12 menambahkan nilai dari variabel wtf ke dalam temp, kemudian menambahkan nilai dari variabel temp ke dalam weight_tf.
5. Baris 13 mengembalikan nilai dari weight_tf.

### 5.3.4 Document Frequency (DF) dan Inverse Document Frequency (IDF)

Proses selanjutnya adalah menghitung nilai DF dan IDF. Nilai TF didapat dengan cara menghitung munculnya token di tiap dokumen, kemudian menghitung nilai dari IDF dengan menggunakan rumus yang telah dijabarkan pada Persamaan 2.2. Implementasi pada proses menghitung nilai DF dan IDF telah disajikan pada Kode Program 5.10.

```
1   def idf(weightedTF):
2     df = []
3     idf = []
4     for i in range(len(weightedTF)):
5       temp = 0
6       for j in range(len(weightedTF[i])):
7         if weightedTF[i][j] > 0:
8           temp +=1
9       df.append(temp)
10    for i in range(len(df)):
11      temp = math.log10((len(weightedTF[i])/df[i]))
12      idf.append(temp)
13    return idf
```

**Kode Program 5.10 Implementasi DF dan IDF**

Penjelasan Kode Program 5.10:

1. Baris 2-3 membuat list bernama “df” dan “idf”.
2. Baris 5 menginisialisasi nilai dari variabel bernama “temp”
3. Baris 6-9 melakukan seleksi kondisi. Apabila nilai dari wtf suatu term bernilai diatas nol, maka nilai dari variabel temp bertambah satu, kemudian menambahkan nilai dari variabel temp ke dalam list df.
4. Baris 10-12 mengimplementasikan formula idf dan menyimpan nilai tersebut ke dalam variabel temp, kemudian menyimpan nilai dari variabel temp ke dalam list idf.
5. Baris 13 mengembalikan nilai dari idf.

### 5.3.5 Term Frequency-Inverse Index Frequency (TF-IDF)

Tahap terakhir adalah menghitung TF-IDF. Nilai TF-IDF didapat dengan cara menggunakan rumus yang telah dijabarkan pada Persamaan 2.3. Implementasi pada proses menghitung nilai TF-IDF dapat dilihat pada Kode Program 5.11.

```
1  def tf_idf(idf, weightedTF):
2      tf-idf = []
3      for i in range(len(idf)):
4          tf_idf = []
5          for j in range(len(weightedTF[i])):
6              if weightedTF[i][j] >0:
7                  temp = idf[i] * weightedTF[i][j]
8                  tf_idf.append(temp)
9              else:
10                 tf_idf.append(0)
11         tf-idf.append(tf_idf)
12     return tf-idf
```

**Kode Program 5.11 Implementasi TF-IDF**

Penjelasan Kode Program 5.11:

1. Baris 2 membuat list bernama "tf-idf".
2. Baris 4 membuat list bernama "tf_idf".
3. Baris 6-10 melakukan seleksi kondisi. Apabila nilai dari wtf suatu term bernilai di atas nol, maka dilakukan operasi aritmetika berupa perkalian dari variabel idf dengan variabel wtf. Setelah itu menyimpan nilainya ke dalam variabel temp. Kemudian menambahkan nilai dari variabel temp ke dalam list tf_idf. Namun, jika nilai dari variabel wtf nol, nilai nol langsung ditambahkan ke dalam list tf_idf.
4. Baris 11 menambahkan nilai dari list tf_idf ke dalam list tf-idf.
5. Baris 12 mengembalikan nilai dari tf-idf.

## 5.4 *Lexicon Based Feature*

Tahap ketiga implementasi pada penelitian ini adalah melakukan pembobotan berdasarkan *lexicon* atau bisa disebut dengan *Lexicon Based Feature*. Pada tahapan ini, dibutuhkan kamus kata positif, kamus kata negatif, kamus besar Bahasa Indonesia, dan kamus kata penegasan untuk dapat menghitung bobot F1 hingga F15.

### 5.4.1 Jumlah Kata Penegasan (F1)

Tahapan ini dilakukan untuk menghitung banyaknya kata penegasan (F1) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F1 telah disajikan pada Kode Program 5.12.

```
1  def booster(text, booster_list):
2      f1 = []
3      for i in range(len(text)):
4          temp = 0
5          for n in text[i]:
6              if n in booster_list:
7                  temp +=1
8          f1.append(temp)
9      return f1
```

**Kode Program 5.12 Implementasi F1**

Penjelasan Kode Program 5.12:

1. Baris 2 membuat list bernama “f1”.
2. Baris 4 menginisialisasi nilai dari variabel temp.
3. Baris 5-7 melakukan perulangan pada tiap dokumen dan pada tiap kata pada dokumen, kemudian melakukan seleksi kondisi. Apabila kata terdapat pada “booster_list”, maka nilai dari variabel temp ditambah dengan satu.
4. Baris 8 menambahkan nilai dari variabel temp ke dalam list f1.
5. Baris 9 mengembalikan nilai dari f1.

### 5.4.2 Jumlah Kata Positif (F2) dan Kata Negatif (F3)

Tahapan ini dilakukan untuk menghitung jumlah kata positif (F2) dan kata negatif (F3) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F2 dan F3 telah disajikan pada Kode Program 5.13.

```
1  def posneg(text, kamus_positif, kamus_negatif):
2    f2 = []
3    f3 = []
4    for i in range(len(text)):
5      temp1 = 0
6      temp2 = 0
7      for n in text[i]:
8        if n in kamus_positif:
9          temp1 +=1
10       if n in kamus_negatif:
11         temp2 +=1
12     f2.append(temp1)
13     f3.append(temp2)
14   return f2,f3
```

**Kode Program 5.13 Implementasi F2 dan F3**

Penjelasan Kode Program 5.13:

1. Baris 2-3 membuat list bernama “f2” dan “f3”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.
3. Baris 7-11 melakukan perulangan pada tiap dokumen dan pada tiap kata pada dokumen dan melakukan seleksi kondisi apabila kata terdapat pada list “kamus_positif” maka nilai dari variabel temp1 ditambah satu, lalu apabila kata terdapat pada list “kamus_negatif” maka nilai dari variabel temp2 ditambah satu.
4. Baris 12-13 menambahkan nilai dari variabel temp1 ke dalam list f2 dan nilai dari temp2 ke dalam list f3.
5. Baris 14 mengembalikan nilai dari f2, f3.

### 5.4.3 Jumlah Kata Sifat Positif (F4) dan Kata Sifat Negatif (F5)

Tahapan ini dilakukan untuk menghitung jumlah kata sifat positif (F4) dan kata sifat negatif (F5) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F4 dan F5 telah diuraikan pada Kode Program 5.14.

```
1  def adjposneg(text, kamuspos, kamusneg, kbbi_list):
2      f4 = []
3      f5 = []
4      for i in range(len(text)):
5          temp1 = 0
6          temp2 = 0
7          for n in text[i]:
8              if n in kbbi_list:
9                  if kbbi_list[n] == 'Adjektiva':
10                     if n in kamuspos:
11                         temp1 += 1
12                     if n in kamusneg:
13                         temp2 += 1
14         f4.append(temp1)
15         f5.append(temp2)
16     return f4,f5
```

**Kode Program 5.14 Implementasi F4 dan F5**

Penjelasan Kode Program 5.14:

1. Baris 2-3 membuat list bernama “f4” dan “f5”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.
3. Baris 7-13 melakukan perulangan pada tiap dokumen dan pada tiap kata pada dokumen dan melakukan seleksi kondisi, apabila kata terdapat pada list “kamuspos” maka nilai dari variabel temp1 ditambah satu, lalu apabila kata terdapat pada list “kamusneg” maka nilai dari variabel temp2 ditambah satu.
4. Baris 13-14 menambahkan nilai dari variabel temp1 ke dalam list f4 dan nilai dari temp2 ke dalam list f5.
5. Baris 14 mengembalikan nilai dari f4, f5.

### 5.4.4 Jumlah Kata Kerja Positif (F6) dan Kata Kerja Negatif (F7)

Tahapan ini dilakukan untuk menghitung jumlah kata kerja positif (F6) dan kata kerja negatif (F7) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F6 dan F7 telah diuraikan pada Kode Program 5.15.

```
1  def verbposneg(text, kamuspos, kamusneg, kbbi_list):
2      f6 = []
3      f7 = []
4      for i in range(len(text)):
5          temp1 = 0
6          temp2 = 0
7          for n in text[i]:
8              if n in kbbi_list:
9                  if kbbi_list[n] == 'Verba':
10                     if n in kamuspos:
11                         temp1 += 1
12                     if n in kamusneg:
13                         temp2 += 1
14         f6.append(temp1)
15         f7.append(temp2)
16     return f6,f7
```

**Kode Program 5.15 Implementasi F6 dan F7**

Penjelasan Kode Program 5.15:

1. Baris 2-3 membuat list bernama “f6” dan “f7”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.
3. Baris 7-13 melakukan perulangan pada tiap dokumen dan pada tiap kata pada dokumen dan melakukan seleksi kondisi apabila kata terdapat pada kamus “kbbi_list” dengan *value* “Verba” maka dilakukan seleksi kondisi. Apabila kata terdapat pada list “kamuspos” maka nilai dari variabel temp1 ditambah satu, lalu apabila kata terdapat pada list “kamusneg” maka nilai dari variabel temp2 ditambah satu.
4. Baris 13-14 menambahkan nilai dari variabel temp1 ke dalam list f6 dan nilai dari temp2 ke dalam list f7.
5. Baris 14 mengembalikan nilai dari f6, f7.

### 5.4.5 Jumlah Kata Keterangan Positif (F8) dan Kata Keterangan Negatif (F9)

Tahapan ini dilakukan untuk menghitung jumlah kata keterangan positif (F8) dan kata keterangan negatif (F9) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F8 dan F9 telah diuraikan pada Kode Program 5.16.

```
1   def advposneg(text, kamuspos, kamusneg, kbbi_list):
2     f8 = []
3     f9 = []
4     for i in range(len(text)):
5       temp1 = 0
6       temp2 = 0
7       for n in text[i]:
8         if n in kbbi_list:
9           if kbbi_list[n] == 'Adverbia':
10            if n in kamuspos:
11              temp1 += 1
12            if n in kamusneg:
13              temp2 += 1
14      f8.append(temp1)
15      f9.append(temp2)
16    return f8,f9
```

**Kode Program 5.16 Implementasi F8 dan F9**

Penjelasan Kode Program 5.16:

1. Baris 2-3 membuat list bernama “f8” dan “f9”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.
3. Baris 7-13 melakukan perulangan pada tiap dokumen dan pada tiap kata pada dokumen dan melakukan seleksi kondisi. Apabila kata terdapat pada kamus “kbbi_list” dengan *value* “Adverbia” maka dilakukan seleksi kondisi. Apabila kata terdapat pada list “kamuspos” maka nilai dari variabel temp1 ditambah satu. Lalu apabila kata terdapat pada list “kamusneg” maka nilai dari variabel temp2 ditambah satu.

4. Baris 13-14 menambahkan nilai dari variabel temp1 ke dalam list f8 dan nilai dari temp2 ke dalam list f9.
5. Baris 14 mengembalikan nilai dari f8, f9.

### 5.4.6 Persentase Kata Sifat Positif (F10) dan Kata Sifat Negatif (F11)

Tahapan ini dilakukan untuk menghitung persentase kata sifat positif (F10) dan kata sifat negatif (F11) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F10 dan F11 telah diuraikan pada Kode Program 5.17.

```
1   def adjpercen(text, f2, f3, f4, f5):
2     f10 = []
3     f11 = []
4     for i in range(len(text)):
5       temp1 = 0
6       temp2 = 0
7       if f4[i] > 0:
8         temp1 = f4[i]/f2[i]
9       if f5[i] > 0:
10        temp2 = f5[i]/f3[i]
11      f10.append(temp1)
12      f11.append(temp2)
13    return f10,f11
```

**Kode Program 5.17 Implementasi F10 dan F11**

Penjelasan Kode Program 5.17:

1. Baris 2-3 membuat list bernama “f10” dan “f11”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.
3. Baris 7-8 melakukan seleksi kondisi. Apabila nilai dari F4 pada suatu dokumen lebih dari nol, maka dilakukan operasi pembagian dengan membagi nilai dari f4 dengan nilai dari f2 pada suatu dokumen kemudian hasil dari operasi pembagian tersebut disimpan ke dalam variabel temp1.
4. Baris 9-10 melakukan seleksi kondisi. Apabila nilai dari F5 pada suatu dokumen lebih dari nol, maka dilakukan operasi pembagian dengan membagi nilai dari f5 dengan nilai dari f3 pada suatu dokumen kemudian hasil dari operasi pembagian tersebut disimpan ke dalam variabel temp2.
5. Baris 11-12 menambahkan nilai dari variabel temp1 ke dalam list f10 dan nilai dari temp2 ke dalam list f11.
6. Baris 14 mengembalikan nilai dari f10, f11.

### 5.4.7 Persentase Kata Kerja Positif (F12) dan Kata Kerja Negatif (F13)

Tahapan ini dilakukan untuk menghitung persentase kata kerja positif (F12) dan kata kerja negatif (F13) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F12 dan F13 telah diuraikan pada Kode Program 5.18.

```
1   def verbpercen(text, f2, f3, f6, f7):
2     f12 = []
3     f13 = []
4     for i in range(len(text)):
5       temp1 = 0
```

```
6       temp2 = 0
7       if f6[i] > 0:
8           temp1 = f6[i]/f2[i]
9       if f7[i] > 0:
10          temp2 = f7[i]/f3[i]
11      f12.append(temp1)
12      f13.append(temp2)
13    return f12,f13
```

**Kode Program 5.18 Implementasi F12 dan F13**

Penjelasan Kode Program 5.18:

1. Baris 2-3 membuat list bernama “f12” dan “f13”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.
3. Baris 7-8 melakukan seleksi kondisi. Apabila nilai dari F6 pada suatu dokumen lebih dari nol, maka dilakukan operasi pembagian dengan membagi nilai dari f6 dengan nilai dari f2 pada suatu dokumen kemudian hasil dari operasi pembagian tersebut disimpan ke dalam variabel temp1.
4. Baris 9-10 melakukan seleksi kondisi. Apabila nilai dari F7 pada suatu dokumen lebih dari nol, maka dilakukan operasi pembagian dengan membagi nilai dari f7 dengan nilai dari f3 pada suatu dokumen kemudian hasil dari operasi pembagian tersebut disimpan ke dalam variabel temp2.
5. Baris 11-12 menambahkan nilai dari variabel temp1 ke dalam list f12 dan nilai dari temp2 ke dalam list f13.
6. Baris 14 mengembalikan nilai dari f12, f13.

### 5.4.8 Persentase Kata Keterangan Positif (F14) dan Kata Keterangan Negatif (F15)

Tahapan ini dilakukan untuk menghitung persentase kata kerja positif (F14) dan kata kerja negatif (F15) yang ada pada tiap dokumen. Implementasi perhitungan jumlah F14 dan F15 telah diuraikan pada Kode Program 5.19.

```
1   def advpercen(text, f2, f3, f8, f9):
2     f14 = []
3     f15 = []
4     for i in range(len(text)):
5       temp1 = 0
6       temp2 = 0
7       if f8[i] > 0:
8           temp1 = f8[i]/f2[i]
9       if f9[i] > 0:
10          temp2 = f9[i]/f3[i]
11      f14.append(temp1)
12      f15.append(temp2)
13    return f14,f15
```

**Kode Program 5.19 Implementasi F14 dan F15**

Penjelasan Kode Program 5.19:

1. Baris 2-3 membuat list bernama “f14” dan “f15”.
2. Baris 5-6 menginisialisasi nilai dari variabel temp1 dan temp2.

3. Baris 7-8 melakukan seleksi kondisi. Apabila nilai dari F8 pada suatu dokumen lebih dari nol, maka dilakukan operasi pembagian dengan membagi nilai dari f8 dengan nilai dari f2 pada suatu dokumen kemudian hasil dari operasi pembagian tersebut disimpan ke dalam variabel temp1.
4. Baris 9-10 melakukan seleksi kondisi. Apabila nilai dari F8 pada suatu dokumen lebih dari nol, maka dilakukan operasi pembagian dengan membagi nilai dari f8 dengan nilai dari f3 pada suatu dokumen kemudian hasil dari operasi pembagian tersebut disimpan ke dalam variabel temp2.
5. Baris 11-12 menambahkan nilai dari variabel temp1 ke dalam list f14 dan nilai dari temp2 ke dalam list f15.
6. Baris 14 mengembalikan nilai dari f14, f15.

## 5.5 Support Vector Machine

Setelah mendapatkan mendapatkan bobot TF-IDF dan *Lexicon Based Feature,* tahap keempat implementasi pada penelitian ini adalah melakukan implementasi Support Vector Machine untuk mendeteksi *tweet* perundungan. Langkah-langkah pada Support Vector Machine adalah menghitung Kernel RBF, menghitung matriks Hessian, *Sequential Training*, menghitung nilai Bias, dan terakhir Testing.

### 5.5.1 Kernel RBF

Tahap awal dalam melakukan implementasi Support Vector Machine adalah menghitung Kernel RBF yang digunakan dalam tahap pembentukan matriks Hessian dan tahap Testing. Implementasi Kernel RBF telah diuraikan pada Kode Program 5.20.

```
1   def kernelRBF(matriks, sigma):
2     kernel = []
3     for i in range(len(matriks[0])):
4       temp = []
5       for j in range(len(matriks[i])):
6         jumlah =0
7         for k in range(len(matriks)):
8           jumlah += math.pow(matriks[k][j]-matriks[k][i],2)
9         kerneli = math.exp((-1*(jumlah))/(2* (math.pow(sigma,2))))
10        temp.append(kerneli)
11      kernel.append(temp)
12    return kernel
```

**Kode Program 5.20 Implementasi Kernel RBF**

Penjelasan Kode Program 5.20:

1. Baris 2 membuat list bernama “kernel”.
2. Baris 4 menginisialisasi nilai dari list bernama “temp”.
3. Baris 5-9 menginisialisasi nilai dari variabel bernama “jumlah” lalu melakukan perulangan terhadap panjangnya variabel “matriks” yang berisi nilai fitur-fitur dari tiap dokumen, kemudian melakukan perhitungan dengan rumus yang telah dijabarkan pada Persamaan 2.5.

4. Baris 9 menambahkan nilai dari kerneli ke dalam list temp
5. Baris 10 menambahkan nilai dari list temp ke dalam list kernel
6. Baris 14 mengembalikan nilai dari list kernel.

### 5.5.2 Matriks Hessian

Setelah mendapatkan nilai dari Kernel RBF, tahapan selanjutnya adalah menghitung Matriks Hessian yang digunakan dalam tahap *sequential learning*. Implementasi matriks Hessian dapat dilihat dalam Kode Program 5.21.

```
1  def Hessian(kernel, kelas, lambdaval):
2      matrikshessian = []
3      for i in range(len(kelas)):
4          temp = []
5          for j in range(len(kelas)):
6              hasil   =   (kelas[i]*kelas[j])*((kernel[i][j])   +   (math.pow
   (lambdaval,2)))
7              temp.append(hasil)
8          matrikshessian.append(temp)
9      return matrikshessian
```

**Kode Program 5.21 Implementasi Matriks Hessian**

Penjelasan Kode Program 5.21:

1. Baris 2 membuat list bernama “matrikshessian”.
2. Baris 4 menginisialisasi nilai dari list bernama “temp”.
3. Baris 5-6 melakukan perulangan terhadap variabel “kelas” yang berisi kelas dari tiap dokumen yang ada pada data latih, lalu melakukan operasi aritmetika dan menyimpan nilainya ke dalam variabel bernama “hasil”.
4. Baris 7 menambahkan nilai dari hasil ke dalam list temp.
5. Baris 8 menambahkan nilai dari list temp ke dalam list matrikshessian.
6. Baris 9 mengembalikan nilai dari list matrikshessian.

### *5.5.3 Sequential Learning*

Setelah mendapatkan nilai dari matriks Hessian, tahapan selanjutnya adalah melakukan *sequential learning* yang bertujuan mencari nilai Support Vector (alpha) yang digunakan dalam perhitungan bias dan *testing*. Implementasi *Sequential Learning* telah diuraikan pada Kode Program 5.22.

```
1  def seq_training(Hessian, imax, gamma, varC, varE):
2      alpha = []
3      deltaalpha = []
4      E = []
5      temp_alpha = []
6      for i in range(len(Hessian)):
7          temp_alpha.append(0)
8      alpha.append(temp_alpha)
9      for i in range(imax):
10         temp_E = []
11         temp_delta = []
```

```
12      temp_alpha = []
13      for j in range(len(Hessian)):
14          jml = 0
15          for k in range(len(Hessian[j])):
16              jml = jml + alpha[-1][j] * Hessian[j][k]
17          temp_E.append(jml)
18      E.append(temp_E)
19      for l in range(len(Hessian)):
20          hasil   =   min(max((gamma*(1-E[-1][l])),   (-1*alpha[-1][l])),
21  (varC-alpha[-1][l]))
22          temp_delta.append(hasil)
23      deltaalpha.append(temp_delta)
24      for m in range(len(temp_delta)):
25          hasil = alpha[-1][m] + deltaalpha[-1][m]
26          temp_alpha.append(hasil)
27      alpha.append(temp_alpha)
28      if(max(temp_delta)<varE):
29          break
30  return alpha[-1]
```

**Kode Program 5.22 Implementasi *Sequential Learning***

Penjelasan Kode Program 5.22:

1. Baris 2-5 membuat list bernama “alpha”, “deltaalpha”, “E”, dan “temp_alpha”.
2. Baris 7-8 menginisialisasi nilai awal alpha.
3. Baris 9-18 melakukan perulangan terhadap list “Hessian” yang berisi matriks Hessian, lalu melakukan operasi aritmetika dan menyimpan nilainya ke dalam variabel bernama “jml” lalu menyimpannya ke dalam list E.
4. Baris 19-23 melakukan perulangan terhadap list “Hessian” yang berisi matriks Hessian, lalu melakukan operasi aritmetika kemudian menyimpan nilainya ke dalam variabel bernama “hasil” lalu menyimpannya ke dalam list deltaalpha.
5. Baris 24-27 melakukan perulangan terhadap list temp_delta yang berisi nilai dari deltaalpha, dan melakukan operasi aritmetika dan menyimpannya ke dalam variabel “hasil” kemudian menyimpannya ke dalam list alpha
6. Baris 28-29 seleksi kondisi apabila ada elemen dari temp_delta yang berjumlah kurang dari “varE” maka dilakukan break.
7. Baris 30 mengembalikan nilai dari list alpha elemen terakhir.

### 5.5.4 Bias

Tahapan selanjutnya adalah menghitung nilai bias untuk mengoptimalkan fungsi *hyperplane*. Implementasi bias telah diuraikan pada Kode Program 5.23 .

```
1   def bias(supportvector, kelas, kernel):
2     xPos = 0
3     xNeg = 0
4     for i in range(len(kelas)):
5       if kelas[i] == 1:
6         xPos = max(xPos, supportvector[i])
7       else:
8         xNeg = max(xNeg, supportvector[i])
9     alphaPos = supportvector.index(xPos)
```

```
10    alphaNeg = supportvector.index(xNeg)
11    sigmaPos = 0
12    sigmaNeg = 0
13    kernelPos = []
14    kernelNeg = []
15    for i in range(len(kelas)):
16      temp_kernelPos = []
17      temp_kernelNeg =[]
18      for j in range(len(kernel[i])):
19        pos = kernel[j][alphaPos]
20        neg = kernel[j][alphaNeg]
21          temp_kernelPos.append(pos)
22          temp_kernelNeg.append(neg)
23      kernelPos = temp_kernelPos
24      kernelNeg = temp_kernelNeg
25      hasilPos = supportvector[i]*kelas[i]*kernelPos[i]
26      hasilNeg = supportvector[i]*kelas[i]*kernelNeg[i]
27      sigmaPos= sigmaPos + hasilPos
28      sigmaNeg= sigmaNeg + hasilNeg
29    bias = -0,5*(sigmaPos+sigmaNeg)
30    return bias
```

**Kode Program 5.23 Implementasi Bias**

Penjelasan Kode Program 5.23:

1. Baris 2-3 menginisialisasi variabel bernama “xPos” dan “xNeg”.
2. Baris 4-10 mencari index dengan nilai alpha tertinggi dari dokumen yang dokumen yang memiliki kelas positif dan negatif.
3. Baris 11-28 melakukan perhitungan untuk menghitung sigma dari masing-masing kelas positif dan negatif.
4. Baris 29 melakukan operasi aritmetika lalu memasukkannya ke dalam variabel bias.
5. Baris 30 mengembalikan nilai dari variabel bias.

### *5.5.5 Testing*

Langkah terakhir pada Support Vector Machine adalah *testing*, berfungsi untuk mengategorikan nilai kelas dalam dokumen data uji. Implementasi dari *testing* telah diuraikan pada Kode Program 5.24 dan Gambar 5.1.

```
1   def testing(kernel, kelas, alpha, bias):
2     kelasuji = []
3     sigma = []
4     for i in range(len(kelas),len(kernel)):
5       jml = 0
6       for j in range(len(kelas)):
7         jml = jml +  (kernel[i][j] * kelas[j] * alpha[j])
8       sigma.append(jml)
9     for i in range(len(sigma)):
10      if sigma[i]+bias >=0:
11        kelas = 1
12      else:
13        kelas = -1
14      kelasuji.append(kelas)
15    return kelasuji
```

**Kode Program 5.24 Implementasi *Testing***

Penjelasan Kode Program 5.24:

1. Baris 2-3 membuat list bernama “kelasuji” dan “sigma”.
2. Baris 4 menginisialisasi nilai dari variabel “jml”.
3. Baris 6-8 melakukan perulangan terhadap panjang dari “kelas” yang berisi banyaknya kelas dari dokumen data uji kemudian melakukan operasi aritmetika dan menyimpan hasilnya ke dalam variabel jml, terakhir menyimpan nilai dari variabel jml ke dalam list sigma.
4. Baris 9-14 melakukan pengulangan terhadap panjang dari list “sigma” setelah itu melakukan seleksi kondisi apabila nilai sigma ditambah dengan bias memiliki hasil lebih dari nol maka nilai kelas adalah satu, sebaliknya nilai kelas adalah negatif satu kemudian menambahkan nilai dari variabel kelas ke dalam list kelasuji.
5. Baris 15 mengembalikan nilai dari list kelas uji.

| | tweet_text | Class |
|---|---|---|
| 0 | presiden @Jokowi mengatakan dalam keberagaman ... | 1 |
| 1 | Nak SD aja paham ini UU sampah. Pak @jokowi ot... | -1 |

**Gambar 5.1 Implementasi Testing**

## 5.6 Evaluasi

Tahap terakhir implementasi pada penelitian ini adalah evaluasi yang bertujuan untuk mengukur kemampuan algoritme yang telah dirancang oleh penulis. Tahap ini dilakukan dengan cara menghitung *confusion matrix* dan kemudian menghitung *accuracy, recall, precision,* dan *f-measure*.

### *5.6.1 Confusion Matrix*

Langkah pertama dalam melakukan evaluasi pada penelitian ini adalah dengan membentuk *confusion matrix*. Implementasi *confusion matrix* dapat dilihat pada Kode Program 5.25.

```
1  def ValConf(predict, actual):
2      TP = 0
3      FN = 0
4      FP = 0
5      TN = 0
6      for i in range(len(actual)):
7          if actual[i] == predict[i]:
8              if actual[i] == -1:
9                  TP +=1
10             else:
11                 TN += 1
12         else:
13             if actual[i] == -1:
14                 FN +=1
```

```
15          else:
16              FP +=1
17      return TP, ,TN FN, FP
```

**Kode Program 5.25 Implementasi *Confusion Matrix***

Penjelasan Kode Program 5.25:

1. Baris 2-5 menginisialisasi variabel "TP", "TN", "FN", dan "FP".
2. Baris 6-16 melakukan perulangan terhadap variabel "actual" yang berisi hasil prediksi kelas data uji, dan melakukan seleksi kondisi untuk menentukan nilai TP, FN, FP, dan TN.
3. Baris 17 mengembalikan nilai dari TP,TN, FN, dan FP.

### 5.6.2 *Accuracy*, *Recall*, *Precision*, dan *F-Measure*

Setelah mendapatkan *confusion matrix*, maka dapat dilakukan perhitungan *accuracy, recall, precision,* dan *F-measure*. Implementasi *accuracy, recall, precision,* dan *F-measure* dapat dilihat padan Kode Program 5.26.

```
1   def eval(TP, FN, FP, TN):
2       accuracy = ((TP+TN)/(TP+FN+FP+TN))*100
3       recall = TP/(TP+FN)
4       precision = TP/(TP+FP)
5       fmeasure = 2* ((precision*recall)/(precision+recall))
6       return accuracy, recall, precision, fmeasure
```

**Kode Program 5.26 Implementasi *Accuracy*, *Recall*, *Precision* dan *F-Measure***

Penjelasan Kode Program 5.26:

1. Baris 2 melakukan operasi aritmetika dan menyimpannya ke dalam variabel "accuracy".
2. Baris 3 melakukan operasi aritmetika dan menyimpannya ke dalam variabel "recall".
3. Baris 4 melakukan operasi aritmetika dan menyimpannya ke dalam variabel "precision".
4. Baris 5 melakukan operasi aritmetika dan menyimpannya ke dalam variabel "fmeasure".
5. Baris 6 mengembalikan nilai dari *accuracy, recall, precision,* dan *fmeasure*.

# BAB 6 PENGUJIAN DAN ANALISIS

Bab pengujian dan analisis membahas mengenai berbagai macam pengujian yang dilakukan kepada tiga kernel pada SVM yaitu, Kernel Linear, Polynomial Degree d, dan Kernel RBF dan pengujian terhadap pengaruh penggunaan *Lexicon Based Feature* terhadap hasil evaluasi. Rasio data latih dan data uji pada perbandingan yang dilakukan adalah 70:30. Setiap pengujian dilakukan pencarian hasil evaluasi yang memiliki nilai *accuracy, recall, preicision,* dan *f-measure* yang tinggi pada setiap parameter.

## 6.1 Pengujian Kernel Gaussian RBF

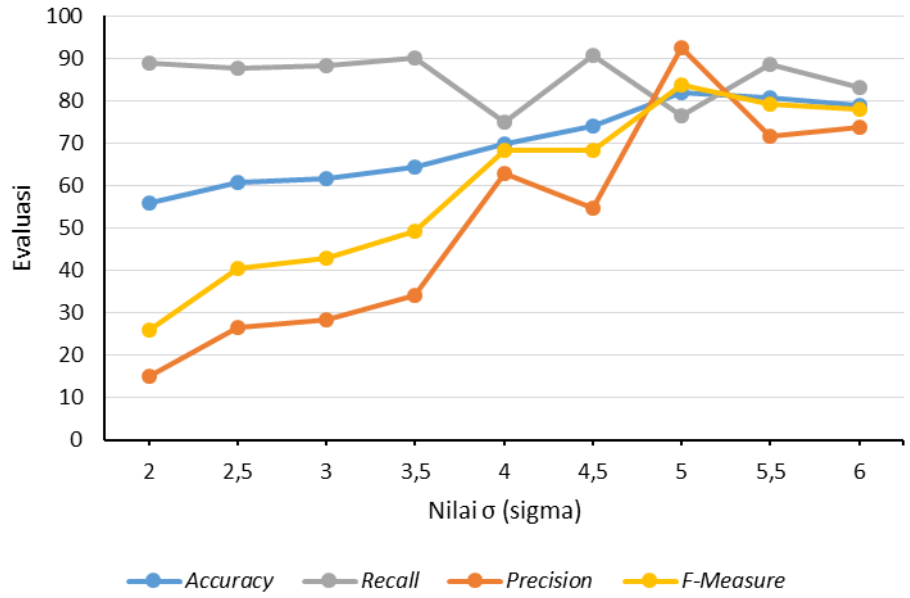
Pada pengujian Kernel *Gaussian RBF* dilakukan enam macam pengujian pada tiap parameternya yaitu, pengujian *Sigma*, *Lambda*, *Gamma*, *Complexity*, *Epsilon*, dan *Iter Max.*

### 6.1.1 Hasil Pengujian *Sigma*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *sigma* terhadap *accuracy*, *recall, precision, dan f-measure*. Pengujian *sigma* menggunakan *lambda* dengan nilai 2, *gamma* dengan nilai 0,001, *complexity* dengan nilai 1, *epsilon* dengan nilai 0,0001, dan *iter max* dengan nilai 40. Hasil pengujian Sigma telah disajikan dalam Tabel 6.1 dan Gambar 6.1.

**Tabel 6.1 Pengujian Sigma**

| Nilai σ (sigma) | Evaluasi | | | |
|---|---|---|---|---|
| | *accuracy* | *Recall* | *Precision* | *F-Measure* |
| 2 | 55,76 | 88,88 | 15,09 | 25,80 |
| 2,5 | 60,57 | 87,5 | 26,41 | 40,57 |
| 3 | 61,53 | 88,23 | 28,30 | 42,85 |
| 3,5 | 64,42 | 90 | 33,96 | 49,31 |
| 4 | 69,76 | 75 | 62,86 | 68,29 |
| 4,5 | 74,03 | 90,62 | 54,71 | 68,23 |
| 5 | 81,73 | 76,56 | 92,45 | 83,76 |
| 5,5 | 80,76 | 88,37 | 71,69 | 79,16 |
| 6 | 78,84 | 82,97 | 73,58 | 78 |




**Gambar 6.1 Grafik Pengujian *Sigma***

Pada Gambar 6.1 dalam pengujian nilai *sigma*, parameter nilai *sigma* yang terbaik ditunjukkan oleh nilai 5, dengan tingkat *accuracy* = 81,73, *recall* = 76,56, *precision* = 92,45, dan *f-measure* = 83,76. Hal ini terjadi karena nilai *sigma* berpengaruh terhadap jarak antar fitur, jika selisih antar fitur lebih besar dari *sigma*, maka nilai pada fungsi kernel mendekati nol dan hal tersebut mengakibatkan hanya fitur pada jarak tertentu yang memengaruhi prediksi pada sistem. Oleh sebab itu, pada grafik pengujian nilai *sigma*, dapat dilihat bahwa nilai evaluasi tidak menyentuh angka 70 sampai nilai dari *sigma* = 4.

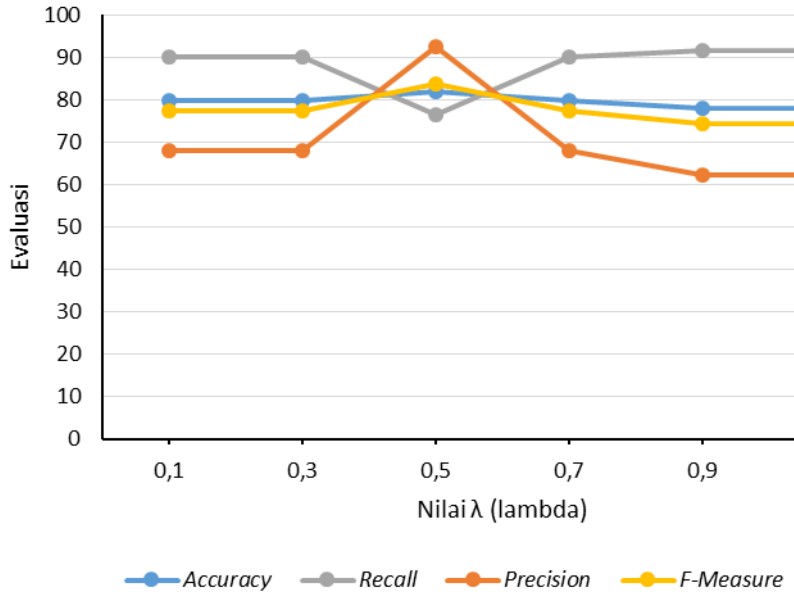
### 6.1.2 Hasil Pengujian *Lambda*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *lambda* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *lambda* menggunakan *sigma* dengan nilai 5, *gamma* dengan nilai 0,0001, *complexity* dengan nilai 1, *epsilon* dengan nilai 0,0001, dan *iter max* dengan nilai 40. Hasil pengujian *lambda* telah disajikan dalam Tabel 6.2 dan Gambar 6.2.

**Tabel 6.2 Tabel Pengujian Parameter λ**

| Nilai λ (lambda) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,1 | 79,80 | 90 | 67,92 | 77,41 |
| 0,3 | 79,80 | 90 | 67,92 | 77,41 |
| 0,5 | 81,73 | 76,56 | 92,45 | 83,76 |
| 0,7 | 79,80 | 90 | 67,92 | 77,41 |
| 0,9 | 77,88 | 91,66 | 62,26 | 74,15 |

| | | | | |
|---|---|---|---|---|
| 1 | 77,88 | 91,66 | 62,26 | 74,15 |
| 2 | 74,03 | 90,62 | 54,71 | 68,23 |
| 3 | 71,15 | 87,09 | 50,94 | 64,28 |




**Gambar 6.2 Grafik Pengujian *Lambda***

Pada Gambar 6.2 dalam pengujian nilai *lambda*, parameter nilai *lambda* yang terbaik ditunjukkan oleh nilai 0,5, dengan tingkat *accuracy* = 81,73*, recall* = 76,56*, precision* = 92,45, dan *f-measure* = 83,76. Hal ini disebabkan karena nilai *lambda* berfungsi untuk mengendalikan nilai dari *error rate* pada *sequential learning.* Semakin besar nilai *lambda*, maka semakin besar pula kemungkinan *overfitting,* dan semakin kecil nilai lambda maka semakin besar kemungkinan *underfitting.*

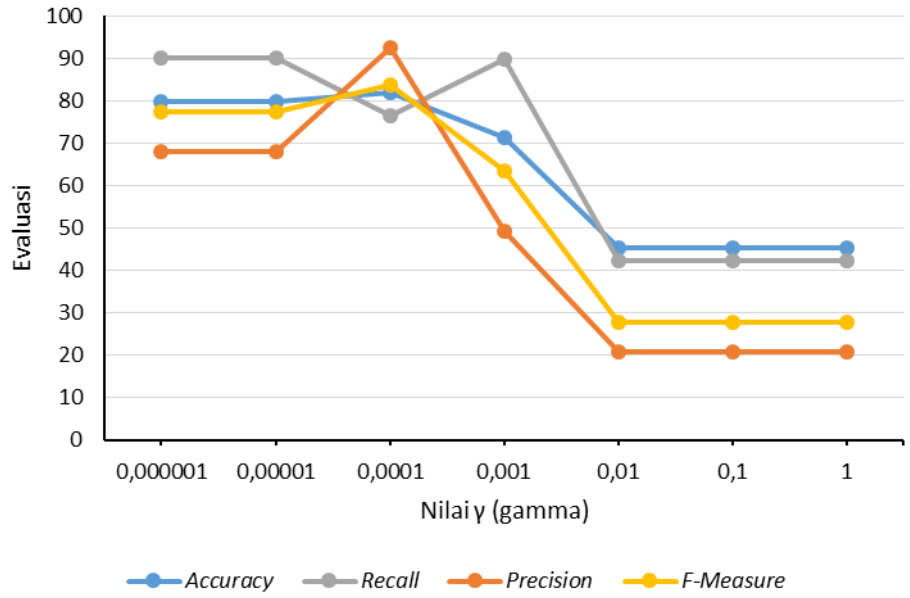
### 6.1.3 Hasil Pengujian *Gamma*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *gamma* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *gamma* menggunakan *sigma* dengan nilai 5, *lambda* dengan nilai 0,5, *complexity* dengan nilai 1, *epsilon* dengan nilai 0,0001, dan *iter max* dengan nilai 40. Hasil pengujian *gamma* telah disajikan dalam Tabel 6.3 dan Gambar 6.3.

**Tabel 6.3 Tabel Pengujian Parameter γ**

| Nilai γ (gamma) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,000001 | 79,80 | 90 | 67,92 | 77,41 |
| 0,00001 | 79,80 | 90 | 67,92 | 77,41 |

| | | | | |
|---|---|---|---|---|
| 0,0001 | 81,73 | 76,56 | 92,45 | 83,76 |
| 0,001 | 71,15 | 89,65 | 49,05 | 63,41 |
| 0,01 | 45,19 | 42,30 | 20,75 | 27,84 |
| 0,1 | 45,19 | 42,30 | 20,75 | 27,84 |
| 1 | 45,19 | 42,30 | 20,75 | 27,84 |




**Gambar 6.3 Grafik Pengujian *Gamma***

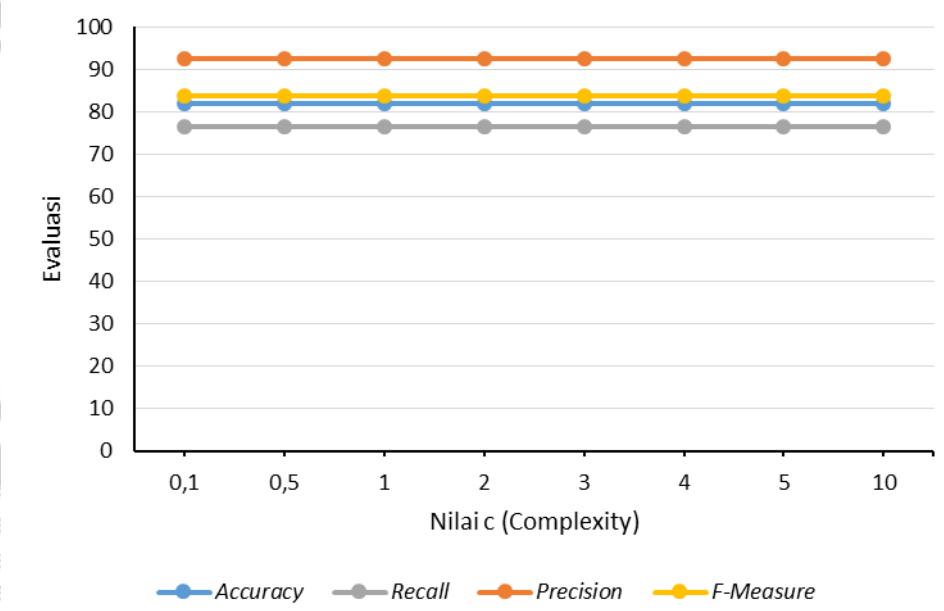
Pada Gambar 6.3 dalam pengujian nilai *gamma*, parameter nilai *gamma* yang terbaik ditunjukkan oleh nilai 0,0001, dengan tingkat *accuracy* = 81,73*, recall* = 76,56*, precision* = 92,45, dan *f-measure* = 83,76. Hasil evaluasi mengalami penurunan drastis pada saat nilai *gamma* 0,01 dengan tingkat *accuracy* = 45,19*, recall* = 42,30*, precision* = 20,75, dan *f-measure* = 27,84. Hal ini dikarenakan nilai *gamma* mengontrol kecepatan pembelajaran pada *sequential learning*, yang berarti semakin besar nilai *gamma* semakin cepat untuk mencapai nilai konvergensi dan semakin besar nilai gamma, semakin mengurangi tingkat ketelitian pada sistem, dan semakin kecil nilai gamma dapat meningkatkan tingkat ketelitian pada sistem.

### 6.1.4 Hasil Pengujian *Complexity*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *complexity* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *complexity* menggunakan *sigma* dengan nilai 5, *lambda* dengan nilai 0,5, *gamma* dengan nilai 0.0001, *epsilon* dengan nilai 0,0001, dan *iter max* dengan nilai 40. Hasil pengujian *complexity* telah disajikan dalam Tabel 6.4 dan Gambar 6.4.

**Tabel 6.4 Tabel Pengujian Parameter c**

| Nilai c (Complexity) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,1 | 81,73 | 76,56 | 92,45 | 83,76 |
| 0,5 | 81,73 | 76,56 | 92,45 | 83,76 |
| 1 | 81,73 | 76,56 | 92,45 | 83,76 |
| 2 | 81,73 | 76,56 | 92,45 | 83,76 |
| 3 | 81,73 | 76,56 | 92,45 | 83,76 |
| 4 | 81,73 | 76,56 | 92,45 | 83,76 |
| 5 | 81,73 | 76,56 | 92,45 | 83,76 |
| 10 | 81,73 | 76,56 | 92,45 | 83,76 |




**Gambar 6.4 Grafik Pengujian *Complexity***

Pada Gambar 6.4 terlihat bahwa tingkat evaluasi tidak berubah terhadap semua nilai parameter C (*complexity*). Hal ini menunjukkan bahwa besar atau kecilnya parameter C tidak berpengaruh terhadap hasil evaluasi.

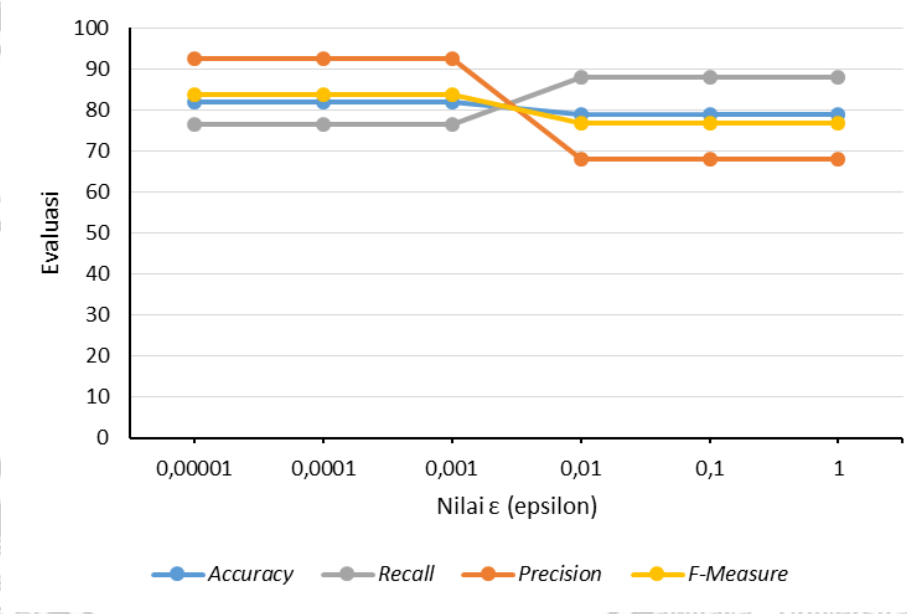
### 6.1.5 Hasil Pengujian *Epsilon*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *epsilon* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *epsilon* menggunakan *sigma* dengan nilai 5, *lambda* dengan nilai 0,5, *gamma* dengan nilai 0.0001, *complexity*

dengan nilai 1, dan *iter max* dengan nilai 40. Hasil Pengujian *epsilon* telah disajikan dalam Tabel 6.5 dan Gambar 6.5.

**Tabel 6.5 Tabel Pengujian Parameter ε**

| Nilai ε (epsilon) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,00001 | 81,73 | 76,56 | 92,45 | 83,76 |
| 0,0001 | 81,73 | 76,56 | 92,45 | 83,76 |
| 0,001 | 81,73 | 76,56 | 92,45 | 83,76 |
| 0,01 | 78,84 | 87,80 | 67,92 | 76,59 |
| 0,1 | 78,84 | 87,80 | 67,92 | 76,59 |
| 1 | 78,84 | 87,80 | 67,92 | 76,59 |




**Gambar 6.5 Grafik Pengujian *Epsilon***

Pada Gambar 6.5 terlihat bahwa nilai evaluasi terbaik ditunjukkan oleh 3 nilai epsilon, yaitu 0,00001, 0,0001, dan 0,001 dengan tingkat *accuracy* = 81,73, *recall* = 76,56, *precision* = 92,45, dan *f-measure* = 83,76. Hasil evaluasi mengalami penurunan pada saat nilai epsilon 0,001, penurunan tersebut terjadi karena epsilon berfungsi sebagai pengontrol laju pertumbuhan *alpha*, dan semakin besar nilai *epsilon*, maka iterasi yang dilakukan pada proses *training* akan semakin sedikit dan mengakibatkan terjadinya konvergensi dini.

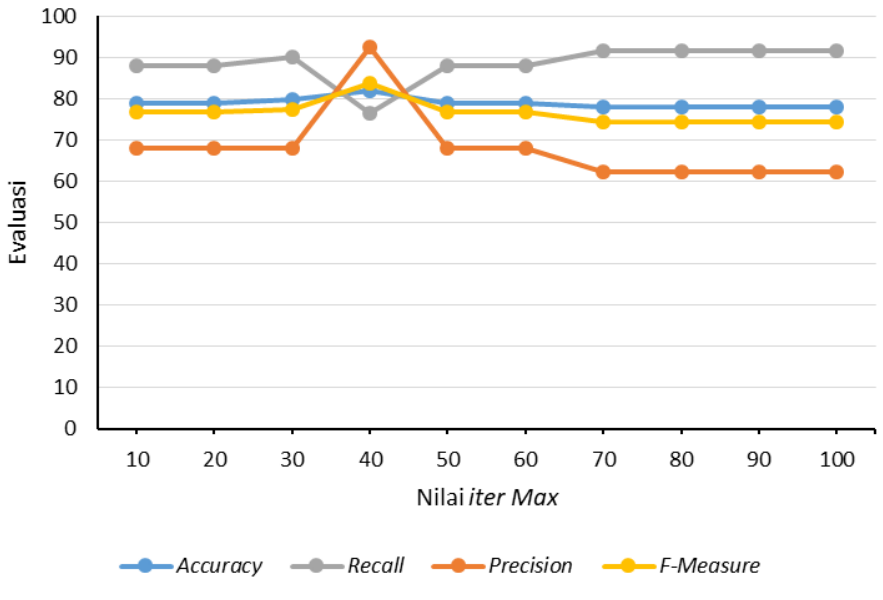
### 6.1.6 Hasil Pengujian *Iter Max*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *iter max* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *iter max* menggunakan *sigma*

dengan nilai 5, *lambda* dengan nilai 0,5, *gamma* dengan nilai 0.0001, *complexity* dengan nilai 1, dan *epsilon* dengan nilai 0.0001. Hasil pengujian *iter max* telah disajikan dalam Tabel 6.6 dan Gambar 6.6.

**Tabel 6.6 Tabel Pengujian Parameter *iter Max***

| Nilai *iter Max* | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 10 | 78,84 | 87,80 | 67,92 | 76,59 |
| 20 | 78,84 | 87,80 | 67,92 | 76,59 |
| 30 | 79,80 | 90 | 67,92 | 77,41 |
| 40 | 81,73 | 76,56 | 92,45 | 83,76 |
| 50 | 78,84 | 87,80 | 67,92 | 76,59 |
| 60 | 78,84 | 87,80 | 67,92 | 76,59 |
| 70 | 77,88 | 91,66 | 62,26 | 74,15 |
| 80 | 77,88 | 91,66 | 62,26 | 74,15 |
| 90 | 77,88 | 91,66 | 62,26 | 74,15 |
| 100 | 77,88 | 91,66 | 62,26 | 74,15 |




**Gambar 6.6 Grafik Pengujian *iter Max***

Pada Gambar 6.6 dalam pengujian nilai *iter max*, parameter nilai *iter max* yang terbaik ditunjukkan oleh nilai 40, dengan tingkat *accuracy* = 81,73, *recall* = 76,56, *precision* = 92,45, dan *f-measure* = 83,76. Nilai *iter max* memengaruhi nilai

dari alpha/Support Vector, Oleh sebab itu, nilai *iter max* yang kecil akan mengakibatkan nilai alpha belum menjadi konvergen, sedangkan nilai *iter max* yang tinggi dapat menyebabkan *overfitting* pada nilai Support Vector.

## 6.2 Pengujian Kernel Linear

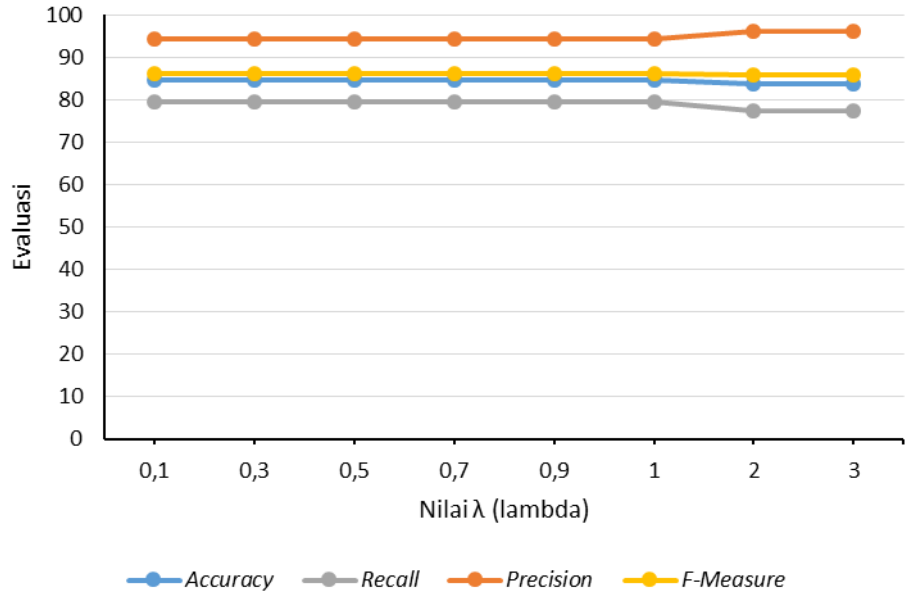
Pada pengujian Kernel Linear dilakukan lima macam pengujian pada tiap parameternya yaitu, pengujian *Lambda*, *Gamma*, *Complexity*, *Epsilon*, dan *Iter Max.*

### 6.2.1 Hasil Pengujian *Lambda*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *lambda* pada *accuracy*, *precision, recall, dan f-measure*. Pengujian *lambda* menggunakan *gamma* dengan nilai 0,0001, *complexity* dengan nilai 1, epsilon dengan nilai 0,0001, dan *iter max* dengan nilai 10. Hasil pengujian *lambda* telah disajikan dalam Tabel 6.7 dan Gambar 6.7.

**Tabel 6.7 Tabel Pengujian Parameter λ**

| Nilai λ (lambda) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,1 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,3 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,5 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,7 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,9 | 84,61 | 79,36 | 94,33 | 86,20 |
| 1 | 84,61 | 79,36 | 94,33 | 86,20 |
| 2 | 83,65 | 77,27 | 96,22 | 85,71 |
| 3 | 83,65 | 77,27 | 96,22 | 85,71 |




**Gambar 6.7 Grafik Pengujian *Lambda***

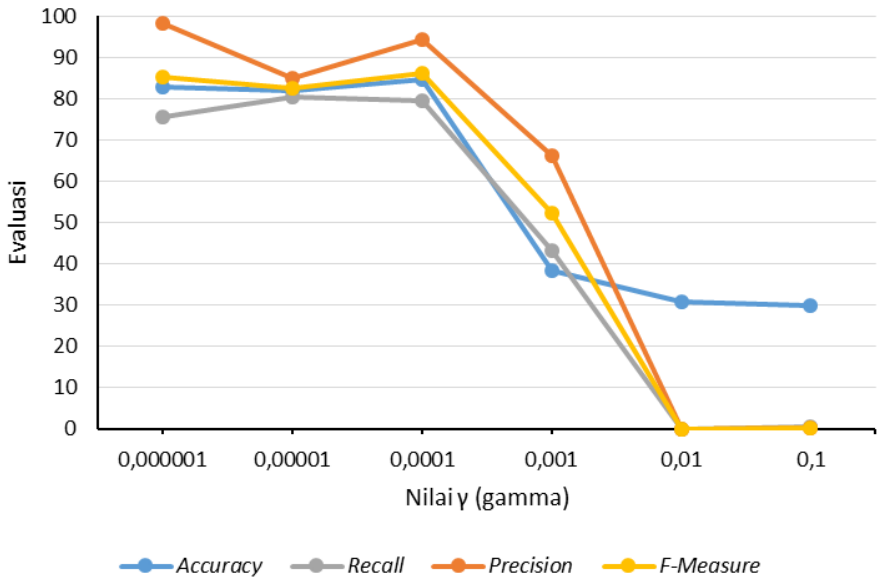
Pada Gambar 6.7 dalam pengujian nilai *lambda*, parameter nilai *lambda* yang terbaik ditunjukkan oleh nilai 0,1 hingga nilai 1, dengan tingkat *accuracy* = 84,61, *recall* = 79,36, *precision* = 94,33, dan *f-measure* = 86,20.

### 6.2.2 Hasil Pengujian *Gamma*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *gamma* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *gamma* menggunakan lambda dengan nilai 0,1, *complexity* dengan nilai 1, epsilon dengan nilai 0,0001, dan *iter max* dengan nilai 10. Hasil pengujian *gamma* telah diajikan dalam Tabel 6.8 dan Gambar 6.8.

**Tabel 6.8 Tabel Pengujian Parameter γ**

| Nilai γ (gamma) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,000001 | 82,69 | 75,36 | 98,11 | 85,24 |
| 0,00001 | 81,73 | 80,35 | 84,90 | 82,56 |
| 0,0001 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,001 | 38,46 | 43,20 | 66,03 | 52,23 |
| 0,01 | 30,76 | 0 | 0 | 0 |
| 0,1 | 29,80 | 0,45 | 0,18 | 0,26 |
| 1 | 28,84 | 0,43 | 0,18 | 0,26 |




**Gambar 6.8 Grafik Pengujian *Gamma***

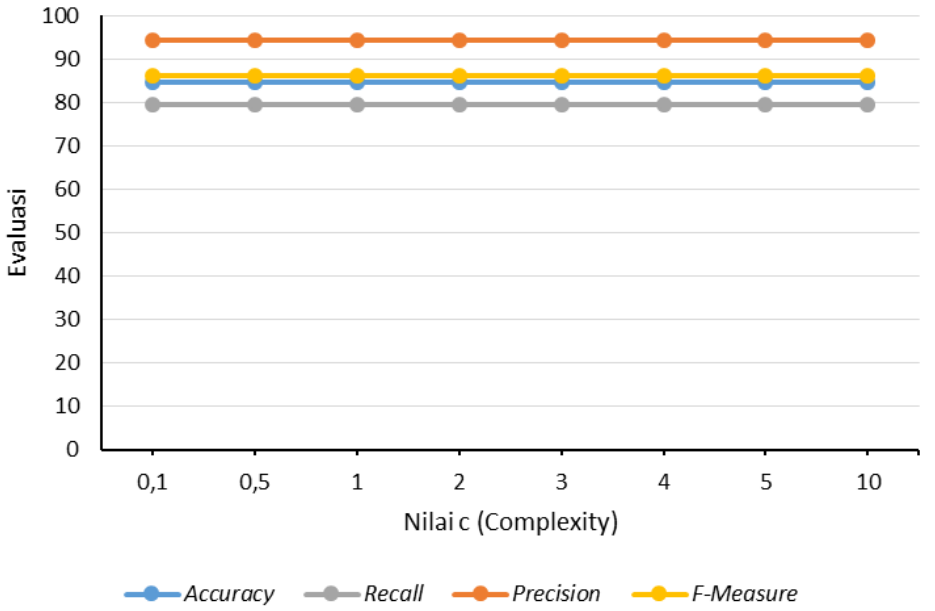
Pada Gambar 6.8 dalam pengujian nilai *gamma*, parameter nilai *gamma* yang terbaik ditunjukkan oleh nilai 0,0001, dengan tingkat *accuracy* = 84,61, *recall* = 79,36, *precision* = 94,33, dan *f-measure* = 86,20.

### 6.2.3 Hasil Pengujian *Complexity*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *complexity* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *complexity* menggunakan *lambda* dengan nilai 0,1, *gamma* dengan nilai 0.0001, *epsilon* dengan nilai 0.0001, dan *iter max* dengan nilai 10. Hasil pengujian *complexity* telah disajikan dalam Tabel 6.9 dan Gambar 6.9.

**Tabel 6.9 Tabel Pengujian Parameter c**

| Nilai c (Complexity) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,1 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,5 | 84,61 | 79,36 | 94,33 | 86,20 |
| 1 | 84,61 | 79,36 | 94,33 | 86,20 |
| 2 | 84,61 | 79,36 | 94,33 | 86,20 |
| 3 | 84,61 | 79,36 | 94,33 | 86,20 |
| 4 | 84,61 | 79,36 | 94,33 | 86,20 |
| 5 | 84,61 | 79,36 | 94,33 | 86,20 |
| 10 | 84,61 | 79,36 | 94,33 | 86,20 |




**Gambar 6.9 Grafik Pengujian *Complexity***

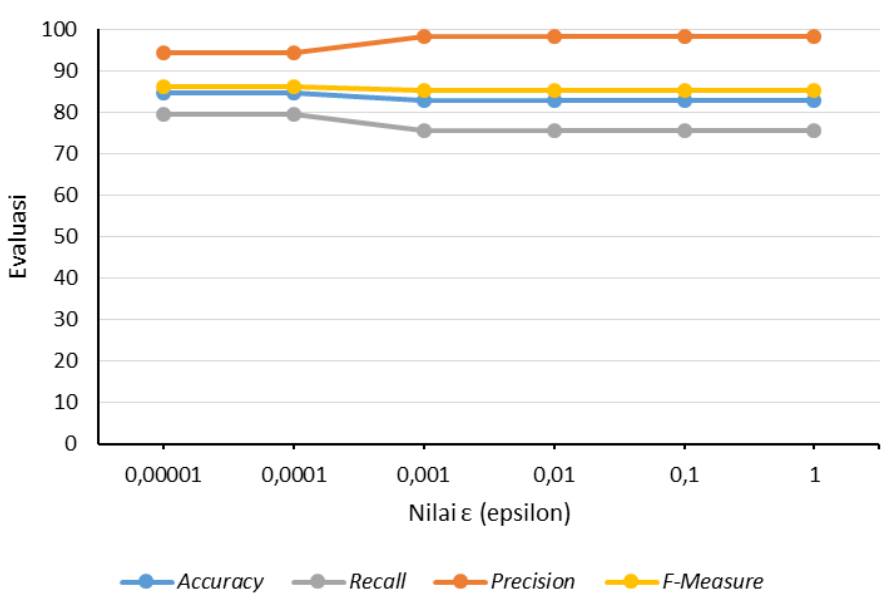
Pada Gambar 6.9 terlihat bahwa tingkat evaluasi tidak berubah terhadap semua nilai parameter C (*complexity*). Hal ini membuktikan bahwa besar atau kecilnya parameter C tidak berpengaruh terhadap hasil evaluasi.

### 6.2.4 Hasil Pengujian *Epsilon*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *epsilon* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *epsilon* menggunakan *lambda* dengan nilai 0,1, gamma dengan nilai 0,0001, *complexity* dengan nilai 1, dan *iter max* dengan nilai 10. Hasil Pengujian *epsilon* telah disajikan dalam Tabel 6.10 dan Gambar 6.10.

**Tabel 6.10 Tabel Pengujian Parameter ε**

| Nilai ε (epsilon) | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 0,00001 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,0001 | 84,61 | 79,36 | 94,33 | 86,20 |
| 0,001 | 82,69 | 75,36 | 98,11 | 85,24 |
| 0,01 | 82,69 | 75,36 | 98,11 | 85,24 |
| 0,1 | 82,69 | 75,36 | 98,11 | 85,24 |
| 1 | 82,69 | 75,36 | 98,11 | 85,24 |




**Gambar 6.10 Grafik Pengujian *Epsilon***

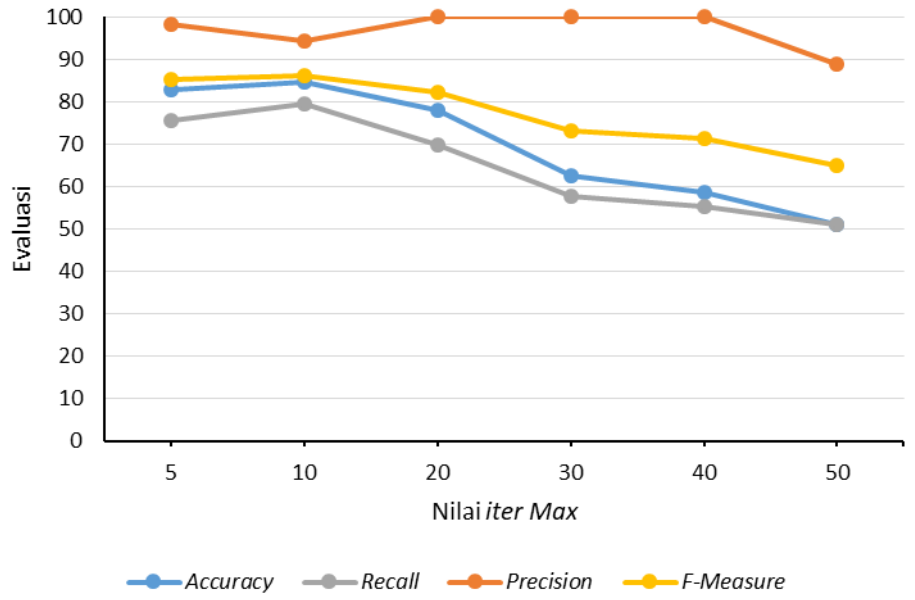
Pada Gambar 6.10 terlihat bahwa nilai evaluasi terbaik ditunjukkan oleh 2 nilai *epsilon*, yaitu 0,00001, dan 0,0001 dengan tingkat *accuracy* = 84,61, *recall* = 79,36, *precision* = 94,33, dan *f-measure* = 86,20. Hasil evaluasi mengalami penurunan pada saat nilai *epsilon* 0,001, penurunan tersebut terjadi karena *epsilon* berfungsi sebagai pengontrol laju pertumbuhan *alpha*, dan semakin besar nilai *epsilon*, maka iterasi yang dilakukan pada proses *training* semakin sedikit dan mengakibatkan terjadinya konvergensi dini.

### 6.2.5 Hasil Pengujian *Iter Max*

Proses ini dilakukan untuk mengukur pengaruh dari parameter *iter max* pada *accuracy*, *recall, precision, dan f-measure*. Pengujian *iter max* menggunakan *lambda* dengan nilai 0,1, *gamma* dengan nilai 0,0001, *complexity* dengan nilai 1, dan *epsilon* dengan nilai 0,0001. Hasil pengujian *iter max* telah disajikan dalam Tabel 6.11 dan Gambar 6.11.

**Tabel 6.11 Tabel Pengujian Parameter *iter Max***

| Nilai *iter Max* | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 5 | 82,69 | 75,36 | 98,11 | 85,24 |
| 10 | 84,61 | 79,36 | 94,33 | 86,20 |
| 20 | 77,88 | 69,73 | 100 | 82,17 |
| 30 | 62,50 | 57,60 | 100 | 73,10 |
| 40 | 58,65 | 55,20 | 100 | 71,14 |
| 50 | 50,96 | 51,08 | 88,67 | 64,82 |




**Gambar 6.11 Grafik Pengujian *iter Max***

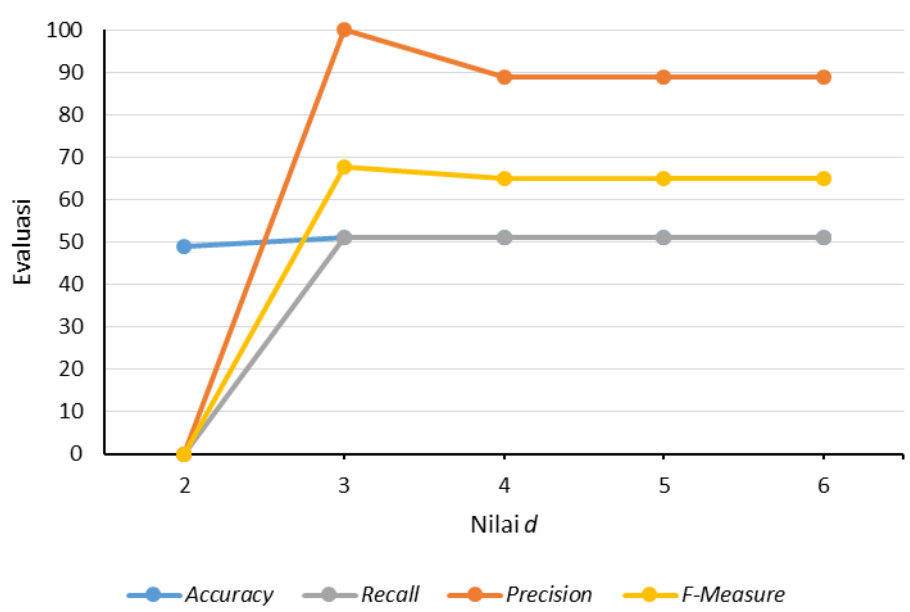
Pada Gambar 6.11 Gambar 6.11 Grafik Pengujian *iter Max* dalam pengujian nilai iter max, parameter nilai *iter max* yang terbaik ditunjukkan oleh nilai 10, dengan tingkat *accuracy* = 84,61, *recall* = 79,36, *precision* = 94,33, dan *f-measure* = 86,20.

## 6.3 Pengujian Kernel Polynomial Degree d

Pada pengujian kernel Polynomial Degree d dilakukan pencarian parameter *d* terbaik, dikarenakan pada Kernel Polynomial Degree d jika menggunakan nilai *d* sama dengan satu membentuk kernel yang sama dengan Kernel Linear, maka parameter pengujian Kernel Polynomial Degree d menggunakan parameter optimal dari Kernel Linear, yaitu *lambda* dengan nilai 0,1, *gamma* dengan nilai 0,0001, *complexity* dengan nilai 1, *epsilon* dengan nilai 0,0001, dan iter max 10. Hasil pengujian kernel Polynomial Degree d telah disajikan dalam Tabel 6.12 dan Gambar 6.12.

**Tabel 6.12 Tabel Pegujian Parameter *d***

| Nilai *d* | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| 2 | 49,03 | 0 | 0 | 0 |
| 3 | 50,96 | 50,96 | 100 | 67,51 |
| 4 | 50,96 | 51,08 | 88,67 | 64,82 |
| 5 | 50,96 | 51,08 | 88,67 | 64,82 |
| 6 | 50,96 | 51,08 | 88,67 | 64,82 |




**Gambar 6.12 Grafik Pengujian nilai *d***

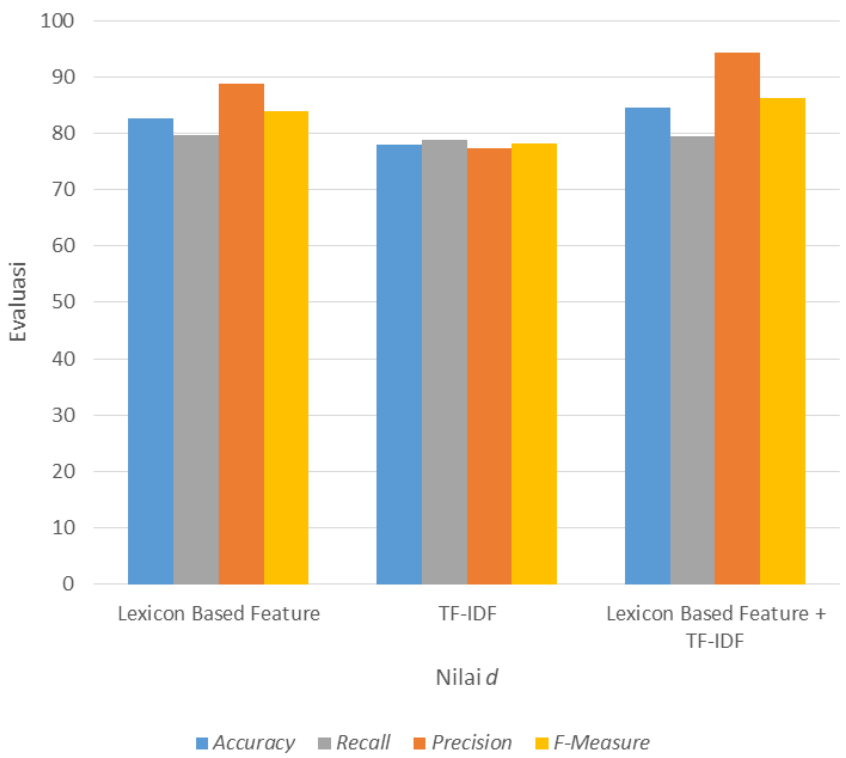
Pada Gambar 6.12 dapat dilihat, hasil evaluasi tertinggi didapatkan oleh d yang bernilai 3 dengan tingkat *accuracy* = 50,96 , *recall* = 50,96, *precision* = 100, dan *f-measure* = 67,51. Setelah itu, hasil evaluasi tidak mengalami peningkatan setelah nilai d bernilai 4 dan seterusnya.

## 6.4 Pengujian *Lexicon Based Feature*

Pada pengujian *Lexicon Based Feature* dilakukan pencarian hasil nilai evaluasi terbaik dengan memakai masing-masing fitur. Dikarenakan pada hasil pengujian kernel, yang memiliki hasil evaluasi terbaik adalah Kernel Linear, maka kernel yang dipakai dalam pengujian ini adalah Linear, dengan menggunakan parameter *lambda* dengan nilai 0,1, *gamma* dengan nilai 0,0001, *complexity* dengan nilai 1 dan *epsilon* dengan nilai 0,0001, dan *iter max* 10. Hasil pengujian *Lexicon Based Feature* telah disajikan dalam Tabel 6.13 dan Gambar 6.13.

**Tabel 6.13 Pengaruh *Lexicon Based Feature***

| Fitur | Evaluasi | | | |
|---|---|---|---|---|
| | *Accuracy* | *Recall* | *Precision* | *F-Measure* |
| *Lexicon Based Feature* | 82,69 | 79,66 | 88,67 | 83,92 |
| TF-IDF | 77,88 | 78,84 | 77,35 | 78,09 |
| *Lexicon Based Feature* + TF-IDF | 84,61 | 79,36 | 94,33 | 86,20 |




**Gambar 6.13 Grafik Pengujian *Lexicon Based Feature***

Pada Gambar 6.13 dapat dilihat, hasil evaluasi menggunakan *Lexicon Based Feature* adalah *accuracy* = 82,69, *recall* = 79,66, *precision* = 88,67, dan *f-measure* = 83,92, hasil evaluasi menggunakan TF-IDF adalah *accuracy* = 77,88, *recall* = 78,84, *precision* = 77,35, dan *f-measure* = 78,09 dan hasil evaluasi fitur gabungan (*Lexicon Based Feature* + TF-IDF) adalah *accuracy* = 84,61 , *recall* = 79,36, *precision* = 94,33, dan *f-measure* = 86,20.

## 6.5 Analisis

Setelah melakukan 4 jenis pengujian, didapatkan hasil bahwa penggunaan Kernel Linear menghasilkan hasil evaluasi yang lebih baik dibanding dengan penggunaan Kernel Gaussian RBF, dan Polynomial Degree d, hal ini terjadi karena kelas pada data dapat dipisahkan secara Linear, sehingga kernel terbaik untuk memisahkan data adalah Kernel Linear. Oleh karena itu, mengubah bentuk Support Vector menjadi dimensi vector space yang lebih tinggi tidak akan meningkatkan hasil evaluasi. Selain itu, terlihat bahwa pada pengujian Kernel Linear, parameter yang berpengaruh terhadap hasil evaluasi adalah parameter *lambda*, *gamma*, *epsilon*, dan *iter max*.

Pada penggunaan fitur TF-IDF, kata-kata pada data tidak mengandung makna secara semantis, dan hanya bergantung terhadap keberadaan fitur pada data latih. Oleh sebab itu, pada data uji "@N0N4m3_90 Dasar manusia pembawa sial!!!!!! @jokowi", sistem gagal mendeteksi perundungan pada data uji tersebut, karena kata "sial" pada data uji tersebut tidak berada dalam fitur.

Pada saat menggunakan *Lexicon Based Feature,* terdapat beberapa kata yang masuk ke dalam kamus positif, tetapi makna sebenarnya dari kata tersebut

adalah negatif sehingga terjadi salah tafsir, sebagai contoh pada data uji "@CNNIndonesia Jgn2 yg menghadiri undangan jg dicopot kewarganegaraanya.... haiiii @jokowi lucu bgt sih lo mimpin negara, peraturan ga tegak sama sekali jadi badut". Kata "lucu" pada data uji tersebut memiliki interpretasi "tidak serius". Akan tetapi, karena kata tersebut berada pada kamus lexicon positif, maka kata tersebut diinterpretasikan sebagai positif, sehingga sistem gagal mendeteksi perundungan pada data uji tersebut.

Penggabungan fitur TF-IDF dengan *Lexicon Based Feature* menghasilkan kinjera sistem yang lebih baik, hal ini dapat dibuktikan pada data uji "@KontraS @amar_grt @jokowi Harusnya yang dievaluasi Plecidennya. Rakyat segera buat Instruksi Rakyat, bila Pleciden menyelengsarakan. . . . Ulangi biar dak diketawai.... Menyelensarakan .... maaf. MENYENGSARAKAN RAKYAT bisa DICOPOT dari Jabatan Pleciden." pada saat menggunakan fitur TF-IDF sistem tidak dapat mendeteksi perundungan, dan pada saat menggunakan *Lexicon Based Feature,* sistem juga tidak dapat mendeteksi perundungan sedangkan apabila kedua fitur digunakan, sistem dapat mendeteksi perundungan pada data uji tersebut. Penggabungan TF-IDF dan *Lexicon Based Feature* berguna untuk menutupi kelemahan tiap fitur karena kata pada data memiliki makna, dan beberapa kata yang tidak terdapat pada kamus lexicon, maupun kata yang salah tafsir dapat teratasi dengan keberadaan kata pada fitur.

Walaupun nilai evaluasi pada fitur gabungan lebih baik daripada fitur penggunaan fitur tunggal. Nilai *recall* pada *Lexicon Based Feature* lebih baik daripada fitur gabungan, hal ini terjadi karena pada *Lexicon Based Feature*, sistem dapat lebih baik menghindari *false negative* karena kata-kata tunggal dimaknai secara semantik. Tabel *confusion matrix* terhadap fitur yang digunakan telah diuraikan dalam Tabel 6.14.

**Tabel 6.14 *Confusion Matrix***

| Fitur | Evaluasi | | | |
|---|---|---|---|---|
| | *TP* | *FP* | *FN* | *TN* |
| *Lexicon Based Feature* | 47 | 6 | 12 | 39 |
| TF-IDF | 41 | 12 | 11 | 40 |
| *Lexicon Based Feature* + TF-IDF | 50 | 3 | 13 | 38 |

Dalam masalah deteksi perundungan lebih baik menekan nilai *false negative* daripada *false positive* karena perundungan akan tetap berdampak pada korban apabila terdapat *tweet* yang mengandung perundungan gagal di deteksi oleh sistem. Nilai *false negative* berpengaruh terhadap tingginya nilai evaluasi *recall*. Semakin kecil nilai *false negative* nilai *recall* akan semakin tinggi. Oleh sebab itu, fitur terbaik dalam masalah deteksi perundungan pada *tweet* adalah *Lexicon Based Feature*, karena memiliki nilai *recall* yang lebih tinggi dari fitur gabungan.

# BAB 7
# PENUTUP

Bab penutup menjelaskan mengenai kesimpulan dan juga saran yang ditujukan kepada penelitian selanjutnya deteksi perundungan siber. Kesimpulan yang didapat berasal dari hasil keseluruhan pekerjaan yang telah dilakukan selama penelitian. Penjelasan kesimpulan dan saran sebagai berikut:

## 7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, didapakan kesimpulan yang berisi inti dari penelitian yang menjawab rumusan masalah sebagai berikut:

1. Penggunaan *Lexicon Based Feature* meningkatkan nilai dari hasil evaluasi dan penggabungan kedua fitur TF-IDF dan *Lexicon Based Feature* menghasilkan nilai evaluasi tertinggi. Namun, nilai evaluasi *recall* dari *Lexicon Based Feature* lebih tinggi dari fitur gabungan, pada masalah deteksi perundungan lebih baik menekan nilai *recall*. Oleh sebab itu, fitur terbaik dalam masalah deteksi perundungan adalah *Lexicon Based Feature*.
2. Hasil evaluasi metode SVM menggunakan *Lexicon Based Feature*, yaitu Kernel Linear dengan parameter *lambda* = 0,1, *gamma* = 0,0001, *complexity* = 1, *epsilon* = 0,0001, dan iter max = 10 menghasilkan *accuracy* = 82,69, *recall* = 79,66, *precision* = 88,67, dan *f-measure* = 83,92.

## 7.2 Saran

Dalam melakukan penelitian ini untuk mendapatkan nilai yang lebih baik atau lebih sempurna, terdapat beberapa hal yang perlu dieksplorasi dan dikembangkan lebih lanjut. Beberapa saran yang dapat diberikan pada penelitian selanjutnya adalah:

1. Menambahkan jumlah dari data latih yang digunakan, agar model algoritme yang dihasilkan lebih baik.
2. Melakukan *K-Fold Cross Validation* agar setiap data yang digunakan muncul pada data latih dan data uji.
3. Melakukan pengujian fitur dari *lexicon* agar mengetahui fitur mana yang lebih memberikan dampak.

# DAFTAR REFERENSI


Abdulloh, N. & Hidayatullah, A. F., 2019. Deteksi Cyberbullying. Volume 1.

Andriansyah, M. et al., 2017. Cyberbullying Comment Classification on Indonesian.

Clement, J., 2019. *statista.* [Online] Available at: https://www.statista.com/statistics/282087/number-of-monthly-active-Twitter-users/ [Accessed 27 3 2020].

Devega, E., 2017. *RI Perangi Konten Negatif.* [Online] Available at: https://www.kominfo.go.id/content/detail/11226/ri-perangi-konten-negatif/0/sorotan_media [Accessed 3 2 2020].

Fachrurrazi, S. & Burhanuddin , 2011. PENGGUNAAN METODE SUPPORT VECTOR MACHINE UNTUK MENGKLASIFIKASI DAN MEMPREDIKSI ANGKUTAN UDARA JENIS PENERBANGAN DOMESTIK DAN PENERBANGAN INTERNASIONAL DI BANDA ACEH.

Feldman, R. & Sanger, J., 2007. *Book ReviewsThe Text Mining Handbook:Advanced Approaches to Analyzing Unstructured Data.* Cambridge: Cambridge University Press.

Gaigole, P. C., Patil, L. H. & Chaudhari, P. M., 2013. Preprocessing Techniques in Text Categorization. *IJCA Proceedings on National Conference on Innovative Paradigms in Engineering & Technology 2013 NCIPET,* pp. 1-3.

Hand, D., Mannila, H. & Smyth, P., 2001. *Principles of Data Mining.* Cambridge: The MIT Press.

Han, J., Kamber, M. & Pei, J., 2011. *Data Mining: Concepts and Techniques.* 3rd ed. Waltham: Morgan Kaufmann.

Hasanah, U., Cholissodin, I. & Fauzi, M. A., 2019. Penentuan Seleksi Atlet Taekwondo Menggunakan Algoritme Support Vector Machine (SVM). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 3, pp. 3580-3588.

Herdiawan, 2016. Analisis Sentimen Terhadap TELKOM Indihome berdasarkan Opini Publik menggunakan Metode Improved K-Nearest Neighbor. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA).*

Kowalski, R. M., Giumetti, G. W., Schroeder, A. M. & Lattaner, M. R., 2014. Bullying in the Digital Age: A Critical Review and Meta-Analysis ofCyberbullying Research Among Youth. *American Psychological Association,* Volume 140, pp. 1073-1137.

Li, B., Yu, S. & Lu, Q., 2003. An Improved k-Nearest Neighbor Algorithm for Text Categorization.

Liu, B., Hu, M. & Cheng, J., 2005. Opinion Observer: Analyzing and Comparing Opinions. *Proceedings of the 14th International World Wide Web Conference (WWW-2005).*

Lunden, I., 2020. *techcrunch.* [Online] Available at: https://techcrunch.com/2020/02/06/Twitter-q4-earnings/ [Accessed 27 3 2020].

Luqyana, W. A., Cholissodin, I. & Perdana, R. S., 2018. Analisis Sentimen Cyberbullying pada Komentar Instagram dengan Metode Klasifikasi Support Vector Machine. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2, pp. 4704-4713.

Manning, C. D., Raghavan, P. & Schütze, H., 2009. *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Mishra, S. & Patil, H., 2017. Improved KNN with Feedback Support. *International Journal of Computer Applications,* Volume 177.

Nathania, D. Z., Indriati & Bachtiar, F. A., 2018. Klasifikasi Spam Pada Twitter Menggunakan Metode Improved K-Nearest Neighbor. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2, pp. 3948-3956.

Noviantho, Isa, S. M. & Ashianti, L., 2017. Cyberbullying Classification using Text Mining. *1st International Conference on Informatics and Computational Sciences (ICICoS),* pp. 241-245.

Nugroho, A. S., Witarto, A. B. & Handoko, D., 2003. Support Vector Machine dan Aplikasinya Dalam Bioinformatika.

Purnamasari, N. M. G. D., Fauzi, M. A., Indriati & Dewi, L. S., 2018. Identifikasi *Tweet* Cyberbullying pada Aplikasi Twitter menggunakan Metode Support Vector Machine (SVM) dan *Information Gain* (IG) sebagai Seleksi Fitur. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2, pp. 5326-5332.

Puspitasari, A. A., Santoso, E. & Indriati, 2018. Klasifikasi Dokumen Tumbuhan Obat Menggunakan Metode Improved k-Nearest Neighbor. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2, pp. 486-492.

Putri, P. A., Ridok, A. & Indriati, 2013. IMPLEMENTASI METODE IMPROVED K-NEAREST NEIGHBOR PADA ANALISIS SENTIMEN TWITTER BERBAHASA INDONESIA. *Doro Jurnal,* Volume 2.

Reynolds, K., Kontostathis, A. & Edwards, L., 2011. Using Machine Learning to Detect Cyberbullying. *10th International Conference on Machine Learning and Applications and Workshops,* Volume 02, pp. 241-244.

Sartana & Afriyeni, N., 2017. PERUNDUNGAN MAYA (CYBER BULLYING) PADA REMAJA AWAL. *JURNAL PSIKOLOGI INSIGHT,* pp. 25-39.

Sianturi, M. H., Marji & Sutrisno, 2014. Perbandingan Kinerja Metode Naive Bayesian Dengan Metode Improved K-NN Dalam Implementasi Sistem Pengklasifikasian Spam Email.

Sidiqua, U. A., Ahsan, T. & Chy, A. N., 2016. Combining a Rule-based Classifier with Ensemble of Feature Sets and Machine Learning Techniques for Sentiment Analysis on Microblog. *International Conference on Computer and Information Technology (ICCIT),* pp. 304-309.

Sticca, F. & Perren, S., 2012. Is cyberbullying worse than traditional bullying? Examining the differential roles of medium, publicity, and anonymity for the perceived severity of bullying.. *Journal of Youth and Adolescence,* Volume 42, p. 739–750.

Syafei, I. & Murfi, H., 2014. Analisis Kinerja Kombinasi Metode Berbasis Lexicon dan Metode Berbasis Learning pada Analisis Sentimen Twitter.

Vijayakumar, S. & Wu, S., 1999. Sequential Support Vector Classifiers and Regression. *IIA/SOCO.*

Wahid, D. H. & SN, A., 2016. Peringkasan Sentimen Esktraktif di Twitter Menggunakan Hybrid TF-IDF dan Cosine Similarity. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems),* Volume 10, pp. 207-218.

Xia, T. & Chai, Y., 2011. An Improvement to TF-IDF: Term Distribution based Term Weight Algorithm. *JOURNAL OF SOFTWARE,* Volume 6, pp. 413-420.

Xu, L., Jiang, C., Wang, J. & Yuan, J., 2014. Information Security in Big Data: Privacy and Data Mining. *IEEE Access,* Volume 2, pp. 1149 - 1176.